D4.4 Final Implementation Report of the Pilot and Future Applications

Document Identification			
Status	Final	**Due Date**	31/12/2021
Version	1.0	**Submission Date**	19/01/2022

Related WP	WP4	**Document Reference**	D4.4
Related Deliverable(s)	D4.3, D4.2, D4.1	**Dissemination Level (*)**	PU
Lead Participant	BUL	**Lead Author**	Derek Groen
Contributors	SZE, BUL, PLUS, DIA, ECMWF, MOON, ARH	**Reviewers**	Dimitris Tsoumakos (ICCS) Javier Nieto (ATOS)

Keywords:
Pilot Applications, Migration, Social Media, Urban Air Pollution, Simulation, High Performance Computing, High Performance Data Analytics.

This document is issued within the frame and for the purpose of the HiDALGO project. This project has received funding from the European Union's Horizon2020 Framework Programme under Grant Agreement No. 824115. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

The dissemination of this document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains. This deliverable is subject to final acceptance by the European Commission.

(*) Dissemination level: **PU**: Public, fully open, e.g. web; **CO:** Confidential, restricted under conditions set out in Model Grant Agreement; **CI:** Classified, **Int =** Internal Working Document, information as referred to in Commission Decision 2001/844/EC.

Document Information

List of Contributors	
Derek Groen	BUL
Alireza Jahani	BUL
Claudio Iacopino	ECMWF
Milana Vuckovic	ECMWF
Diana Suleimenova	BUL
Yani Xue	BUL
Robert Elsaesser	PLUS
Chloe Brimicombe	ECMWF
Amor Messaoud	Moonstar
Zoltán Horváth	SZE

Document History			
0.1	29/06/2021	Alireza Jahani	Structure of the deliverable
0.2	17/11/2021	Derek Groen	ECMWF addition + COVID-19
0.3	18/11/2021	Alireza Jahani	Migration Pilot
0.4	02/12/2021	Robert Elsaesser	Social Media Pilot
0.5	10/12/2021	Derek Groen	Moonstar + other additions
0.6	20/12/2021	Derek Groen	Modified after internal review
1.0	10/01/2022	Derek Groen	Modified after internal review

Quality Control		
Deliverable leader	Derek Groen, BUL	10/01/2022
Quality manager	Marcin Lawenda, PSNC	14/01/2022
Project Coordinator	Francisco Javier Nieto De Santos, ATOS	19/01/2022

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation / acronym	Description
CDR	Call Data Records
CHARM	Brunel hospital bed model
CKAN	Comprehensive Knowledge Archive Network
COST	Cooperation in Science and Technical Development
DRC	Democratic Republic of Congo
Dx.y	Deliverable number y belonging to WP x
EC	European Commission
FACS	The Flu And Coronavirus Simulator
GP	Gaussian process
HPC	High Performance Computing
HPDA	High Performance Data Analytics
ICCS	International Conference on Computational Science
IOM	International Organization for Migration
ITFLOWS	IT Tools and Methods for Managing Migration Flows
LWCC	Largest Weakly Connected Component
MAPE	Mean Average Percentage Error
ML	Machine Learning
MOEA/D	A multiobjective evolutionary algorithm based on decomposition

MUSCLE3	Multiscale Coupling Library and Environment 3
NetCDF	Network Common Data Form
NSGA-II	Non-Dominated Sorting Genetic Algorithm II
NSGA-III	Non-Dominated Sorting Genetic Algorithm III
ORS	Open Route Service
ROM	Reduced Order Model
SEAVEA	Software Environment for Actionable & VVUQ-evaluated Exascale Applications
STAMINA	Sustainable and reliable robotics for part handling in manufacturing
TAC	Temporary Accommodation Centre
TOSCA	Topology and Orchestration Specification for Cloud Applications
UNHCR	United Nations High Commissioner for Refugees
UNOCHA	United Nations Office for the Coordination of Humanitarian Affairs
UTCI	The Universal Thermal Climate Index
WP	Work Package

Executive Summary

This deliverable reports on the final status of the pilot and future applications in HiDALGO. The report, compared to D4.3 [1], contains the latest achievements of the use cases including the Covid-19, migration, urban air pollution, and social networks simulations in response to the main goals and challenges for the 3rd year of the project.

In migration, we present couplings with ECMWF weather data and we present a new AI-based technique that allows Flee to directly give recommendations on the placement of new camps. We also present our work on modelling migration in Tigray, in collaboration with Save The Children. In urban air pollution, we present a new model order reduction approach that saves up to a factor 100 on runtime, and showcase new validation results for the city of Antwerp, among many other things. For social networks, we describe the newly established SN-simulator, present a range of validation results, and study the propagation of Twitter messages for two specific trending topics. We also show that our eigenvalue histogram calculator scales linearly to 10,000s of cores, while our newly developed SN-simulator scales near-linearly to about 500 cores per run. For COVID-19, we present the latest improvements in maturing the rapidly developing Flu And Coronavirus Simulator, and highlight its wide uptake across health organisations throughout Europe.

In terms of coupling we present a wide range of models and data sources that are now integrated with the four pilots. This includes coupling with weather data, telecommunications data, and traffic sensor networks.

Validation and scalability are essential topics within the HiDALGO CoE. As part of this deliverable, we report on the scalability of all pilots on a high level (details can be found in D3.5 [2]) and present validation results for the different pilots in greater detail.

1 Introduction

1.1 Purpose of the document

This deliverable provides insights into the development actions of WP4. Compared to D4.3 [1], this report reflects the development of scopes and uptakes of the pilots and their scientific accomplishments. Besides the Pilot application status, also a detailed assessment about the status of coupled simulations is provided. It also outlines the coupled data sources of the HiDALGO and a detailed assessment about the status of coupled simulations, including weather and climate data sources and models, telecommunication data sources, traffic data and local air quality sensor data, conflict modelling of the migration pilot, and hospitals' bed modelling for the Covid-19.

1.2 Relation to other project work

This deliverable reports the main progress of WP4, and it therefore connects to all technical work packages in the project. The main connections of note are:

- The workflow of the pilots and coupled simulations are designed in collaboration with WP6. While the general design is included in the dedicated WP6 deliverables, we cover the pilot-specific aspects in this deliverable.
- Technical HPC and HPDA tool development, as well as the application-agnostic technical aspects of coupling, and application benchmarking, are presented in WP3. Here we focus mainly on the application-specific developments.
- The HiDALGO infrastructure and portal are developed in the context of WP5. Detailed descriptions about the portal infrastructure are part of the WP5 deliverables.

The results and findings of this work package are analysed with relation to possible collaborations with different stakeholders in WP2, and are disseminated to the general public as well as dedicated interest groups in WP7.

1.3 Structure of the document

This document is structured in 7 major chapters. We introduce the structure of the status reports for the four use cases and the coupled data sources in **Chapter 2.** Then, the status of the pilot applications at M36 of the HiDALGO-project is reported, respectively for migration (**Chapter 3**), Covid-19 simulation (**Chapter 4**) urban air pollution (**Chapter 5**), and social networks (**Chapter 6**). The status of the coupled data sources is reported in **Chapter 7**, and conclusions are drawn in **Chapter 8**.

2 Structure of the status reports on implementation of pilots and coupled data sources

In this section we present the templates that we use for both the pilot and coupling mode / data source descriptions.

2.1 Pilots

For each of the four pilots in HiDALGO (including the COVID-19 pilot which was added in Year 2), we provide a comprehensive status update. This includes:

1. Documentation of the pilot implementation, reporting on the main technical enhancements of the pilot, as well as updates in documentation.
2. Verification and validation status, reporting on what has been done to ensure that the simulations adhere to the theoretical model specifications, and that their results are in accordance with observations.
3. Development of scope and pilot uptake, reporting on any efforts made to extend the applicability of the pilot to new research problems as well as on uptake of the pilot by new external parties.
4. Scaling and performance properties of the pilot use cases, reporting on performance and scalability aspects. The content in this section will be related to goals set in earlier WP4 deliverables (where applicable).
5. Scientific accomplishments, reporting on what each pilot has accomplished in terms of scientific findings and publications.

2.2 Coupled data sources and models

The coupled data sources and models are grouped by the type of coupling. For instance, there is one subsection on integration with ECMWF data and a different one on the Brunel hospital bed model (CHARM).

Within each subsection, we provide an introduction to the coupled data source or model (particularly if it has not been reported on previously), followed by a status report of the coupling mechanisms and a scalability assessment of the coupling.

3 Status of migration pilot

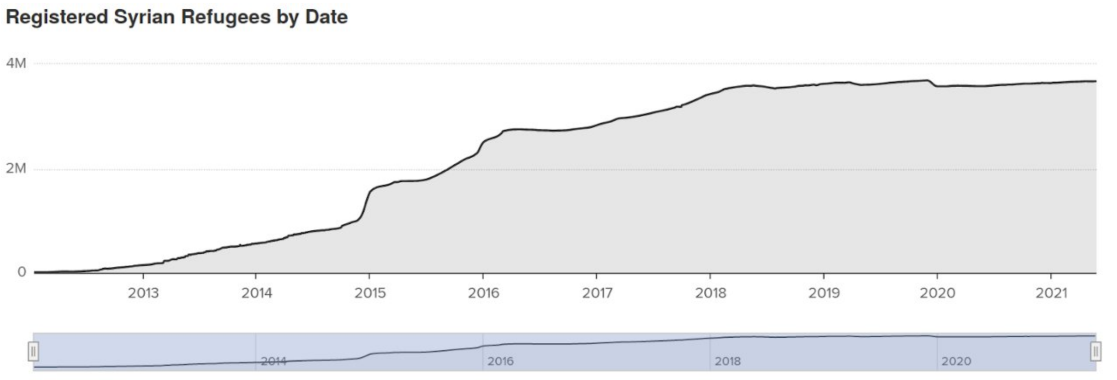
To increase the level of detail in simulation models for the HiDALGO project, we tried to incorporate a broader range of relevant phenomena, such as weather conditions and communications datasets in the migration pilot. Though coupling with communication data has not gone as expected due to data limitations, the incorporation with weather data has been done through different coupling mechanisms and the results show the significance of using additional data sources within migration simulations. Furthermore, in the migration pilot, we aim to improve the scalability and parallelisation, to increase the speed of the application, and to reduce the required time for constructing simulations. As we mentioned in D4.3 [1], we can extend the applicability of our application to different areas like the recent coronavirus pandemic. Therefore, we started a new pilot, COVID-19 simulation pilot, which is explained in more detail in section 4.

3.1 Documentation of the pilot implementation

In this section, we report the overall status of developments on the migration pilot. These include a range of new conflict models, multiscale simulation and coupling with weather data sets, as well as Multi-Objective Optimisation of Camp Location Selection in Forced Migration Scenarios.

Syria Conflict Model

The Syrian Civil War caused millions to flee their homes. As of 2013, 1 in 3 of Syrian refugees (about 667,000 people) sought safety in Lebanon (normally 4.8 million population). Others have fled to Jordan, Turkey, and Iraq. At the time of writing, 5,600,000 Syrian refugees have been registered in Syria's neighbouring countries since the beginning. Figure 1 demonstrates the increasing number of Syrian refugees and asylum-seekers since 2013, which many of them have fled to Turkey, because they believe it is much easier for them to go to the European countries from there. As a result, Turkey has accepted 3,672,646 (May 2021) Syrian refugees, half of whom are spread around cities and a dozen camps placed under the direct authority of the Turkish Government.



Source - Government of Turkey

Figure 1: Refugees and asylum-seekers from Syria - Total (UNHCR, 2021)

To construct the Syria conflict model and simulation, we decided to investigate it in two different time periods. The first period starts from 2013-01-01 and ends in 2016-12-31 (1095 days) and the second period is from 2017-01-01 to 2021-04-20 (1571 days).

First period (2013-2016)

For the first period, which call Syria-2013, we only investigate Syrian conflict at the first administrative level or, in other words, in the provincial level and consider refugees' camps in the neighbouring countries at a country level. For instance, despite having almost 21 refugees' camps in Turkey, we consolidate them all in one camp and name it Turkey. To make it simple and have a general view, we take the same approach for the rest of camps and countries. As a result, in the Syria-2013 model we have 13 conflict locations, 4 camps and 1 town within the location graph (Figure 2).

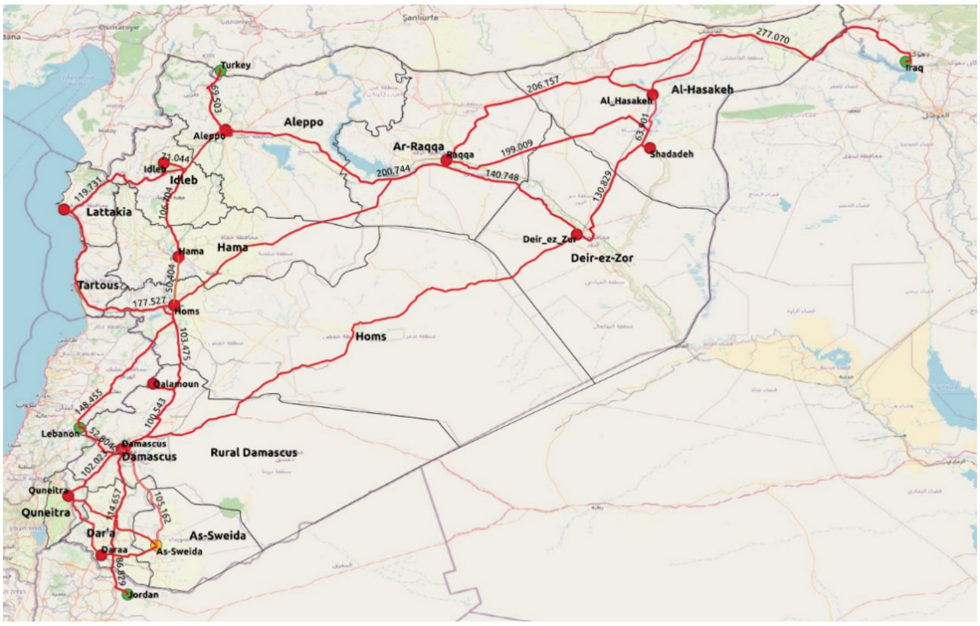


Figure 2: Syria Conflict Location Graph (2013-2016)

Second period (2017-present)

For the second period, which we call Syria-2017 from now on, we investigate Syria at both zero and first administrative levels. In fact, at the administrative_0 level, we add capital cities of Syria provinces as conflict locations and consider refugees' camps in the neighbouring countries at a country level. But, at the administrative_1 level, we decided to include more camps in the neighbouring countries based on their provinces and respectively added more routes to those camps. For instance, we now have three refugees' camps in Turkey. As a result, at the administrative_0 level, we have 15 conflict locations and 4 camps. The blue dots show the rest of other camps which we aggregated their data into the current three camps. (Figure 3).

As reported in Turkey Regional Refugee and Resilience Plan in Response to the Syria Crisis, since 2018, 12 of the 19 Temporary Accommodation Centres (TACs) have been closed following the relocation of Syrians under temporary protection living in the TACs to urban and rural areas or other temporary accommodation centres. As of January 2020, over 98 percent of Syrians under temporary protection lived in urban and rural areas, and less than 2 percent resided in seven TACs that remained open (Altinozu (Hatay), Yayladagi (Hatay), Apaydin (Hatay), Elbeyli (Kilis), Merkez (Adiyaman), Cevdetiye (Osmaniye), Saricam (Adana)).

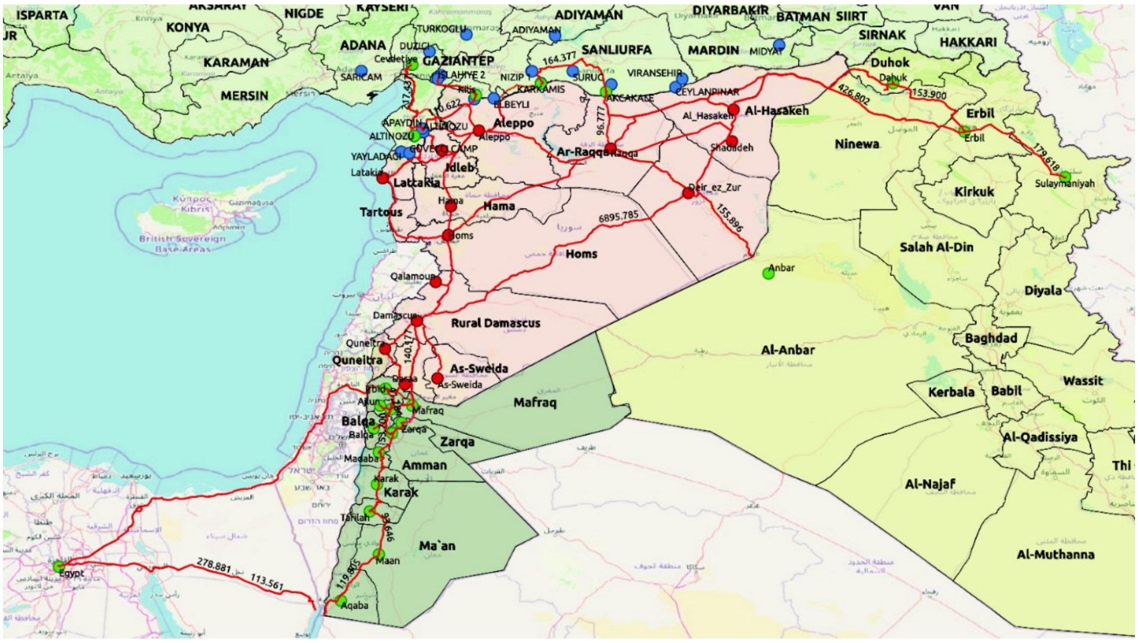


Figure 3: Syria Conflict Location Graph (2017-present) - administrative_1

Mali Conflict Model

The Mali War is an ongoing armed conflict that started in January 2012 between the northern and southern parts of Mali in Africa. Despite the signing of a peace accord two times in 2013 and 2015, low-level fighting continues. Attacks by armed groups that are not parties to the 2015 peace agreement have increased since 2016. Therefore, we decided to construct a new Mali Conflict model and simulate the number of refugees heading to the neighbouring countries. Currently, due to the UNHCR reports, neighbouring countries Burkina Faso, Mauritania and Niger are collectively home to more than 151,500 Malian refugees. Niger currently hosts most of the Malian refugees. Niger hosts more than 61,000 Malian refugees. Mauritania hosts nearly 67,000 and Burkina Faso is home to approximately 23,500.

Figure 4 demonstrates the Mali conflict location graph including conflict locations (Red dots), towns (Yellow dots) and camps (Green dots). In this model, we have 10 conflict locations, 1 town, and 7 camps.

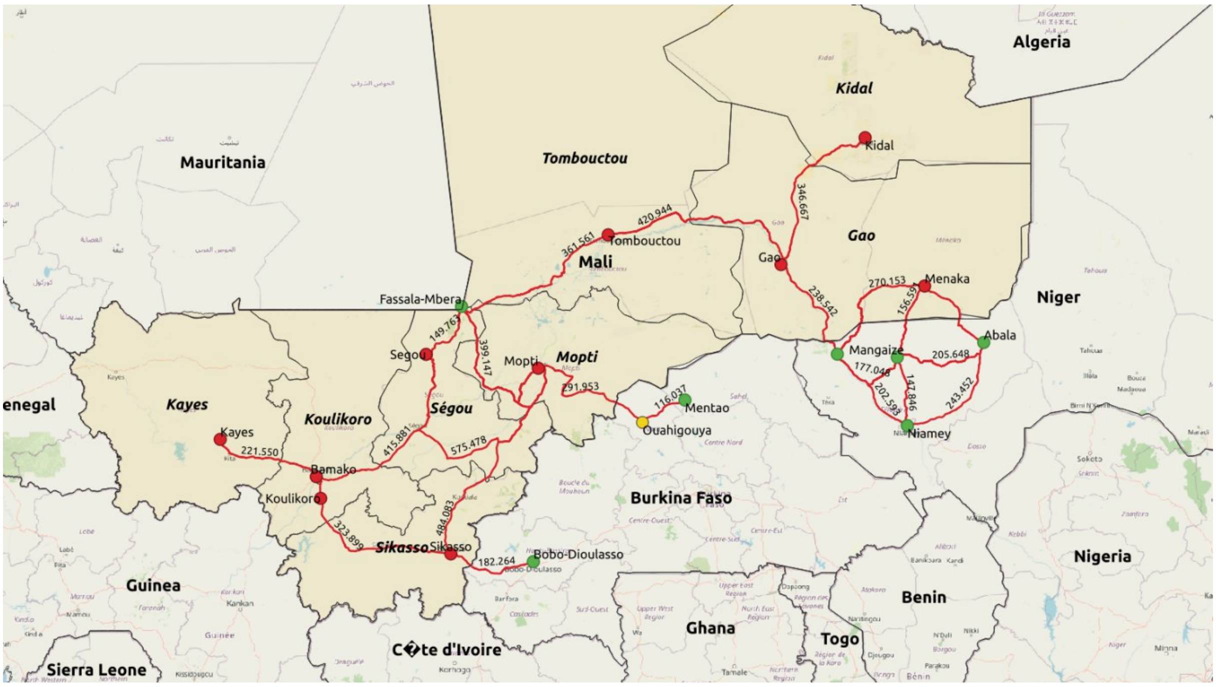


Figure 4: Mali Conflict Location Graph (2016-present)

South Sudan Multiscale Model

To study the South Sudan conflict in more depth and to investigate possible effects of new rules and policy decisions on migration movements (e.g., new link types or new location types), changes in the environment (e.g., weather or river discharge level) and explore our agent-based modelling assumptions, we constructed a multiscale (macroscale and microscale) simulation for the South Sudan conflict. This simulation consists of the following components:

- The macroscale model comprises 8 regions of South Sudan and 14 camps in 4 neighbouring countries, namely Uganda, Kenya, Sudan and Democratic Republic of Congo (DRC).
- The microscale model focuses on forced migrant movements from Upper Nile and Jonglei regions towards Ethiopian camps in Gambela (see Figure 5). We create both models for the same conflict period between 1 June 2016 and 31 July 2017. In the microscale model, we aim to capture key walking routes and roads in mountainous areas in eastern South Sudan. We also increase the level of detail in terms of locations and incorporate a broader range of relevant phenomena, such as weather conditions and food security. Moreover, the Micro model has additional algorithm assumptions. As described before, there are three types of links: drive, walk and river which affect move speed of agents.

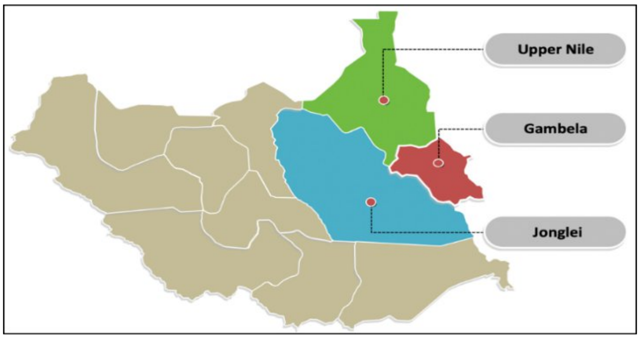


Figure 5: Map of South Sudan. The microscale model includes the Upper Nile, Jonglei and Gambela regions.

More detailed maps of Macro and Micro models' locations are depicted in Figure 6 where in each, red points represent conflict locations, yellow points are towns and green points are camps. Micro model region is shown with a blue background in northeast of South Sudan. To couple the macro and micro models, we use four coupling locations for passing agents between both models. In addition to coupling locations, all Micro model conflict locations should be added to the Macro model as ghost locations. It means that although they are added to the macro model, however they don't have any link to other Macro model locations, and this is why they are called ghosts locations.

All these coupled and ghost locations should be added to separate input file in both models named coupled_locations.csv to make coupling fully working and let agents be spawned between models.

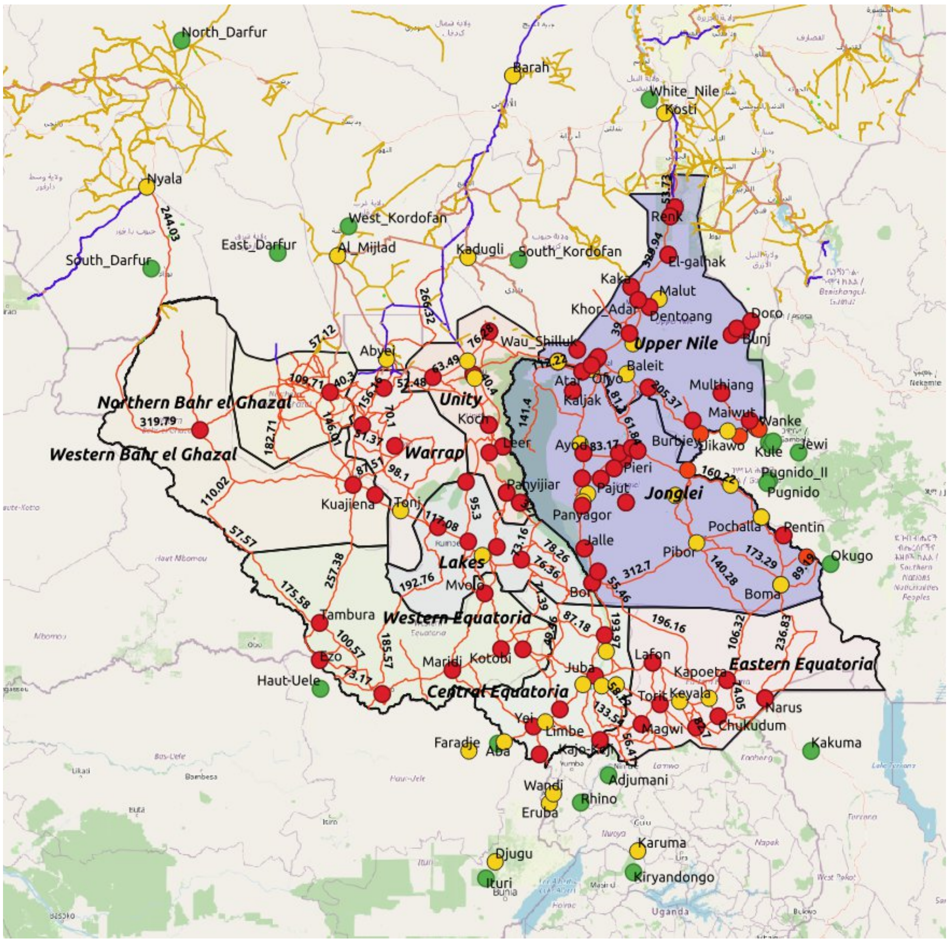


Figure 6: South Sudan location graph indicating macroscale and microscale regions

The results of South Sudan Multiscale simulations with file coupling approach, with and without weather data coupling are presented D3.5 [2].

Multi-Objective Optimisation of Camp Location Selection

Multi-objective optimisation exists in many fields, e.g., automotive design, software engineering, environmental engineering, control engineering, economics, and machine learning. In the Migration pilot, we aim to assist governments and NGOs in the decision making of camp location selection, which can be formulated as a multi-objective optimisation problem. However, it has some difficulties. Multiple criteria/objectives need to be optimised and the constraints related to camp capacity need to be satisfied. Furthermore, the uncertainty related to refugee's movement need to be resolved. Hence, the formulated multi-objective optimisation problem consists of three steps. In the first step, we create a source country with conflict zones and towns, and all possible links among these locations. Second, we add a camp at given coordinates in a destination country. Finally, in the third step, we create a link between the camp and its nearest location in the source country. There are some key assumptions for the model. First, refugees are spawned in the conflict zone(s), second, they move place to place for each time step, according to predefined rules of the Flee, and third, they stop moving once they reach the camp. The objectives of the model are minimising the individual travel distance, maximising the number of successful camp arrivals, and minimising the average idle capacity over the whole simulation days. To solve this problem, we need a population-based metaheuristic algorithm that mimics the natural selection.

Hence, we employed three popular multi-objective evolutionary algorithms to solve camp location selection problem: NSGA-II [3], NSGA-III [4] and MOEA/D [5]. Besides, Flee is used for evaluating objective values of each solution during the evolutionary process. Figure 7 illustrates an example of the route network for the camp location selection problem, which involves one conflict zone, three towns and one refugee camp, and interconnecting roads (lines). The coordinates (x,y) are utilized to indicate locations of a conflict zone, town or refugee camp. The x and y coordinates of the refugee camp are two decision variables of the multi-objective model.

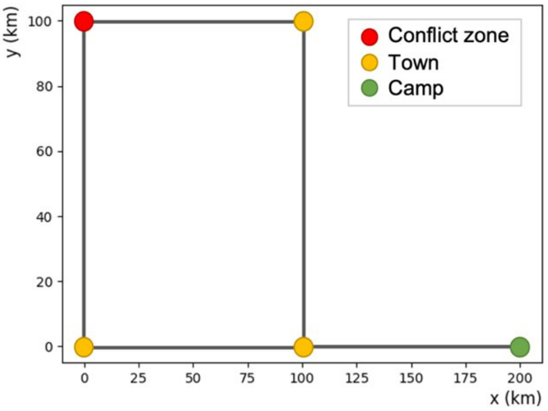


Figure 7: An illustration of route network for the camp location selection problem

As a case study, we used the South Sudan conflict model between 2013 and 2015 for the simulation period of 604 days. In the experiments, we adopted the popular geographic coordinate system WGS 84 for the route network. The variable bound for the geographical coordinates of the camp locations were between 20 to 40 for longitude and 0 and 20 for latitude. To reduce the execution time, we reduced the number of agents by a factor 100. After 6000 runs, we achieved the results shown in Table 1. Figure 8 shows the optimised camp locations (green circles). In summary, among the three tested optimisation algorithms, NSGA-III is the most efficient algorithm and NSGA-II performs the best in terms of providing a set of diverse solutions (optimal camp locations).

Algorithms	Runtime (s)	Longitude	Latitude	Obj_1	Obj_2	Obj_3
NSGA-II	120472.50	31.71067	3.69912	1226.45	811	85.72
		31.66456	3.69912	1143.28	800	77.19
		31.68606	3.66848	1236.42	807	83.24

Algorithms	Runtime (s)	Longitude	Latitude	Obj_1	Obj_2	Obj_3
		30.52305	3.81140	1223.73	800	72.44
		30.52441	3.83310	1193.43	802	75.03
		30.52478	3.82853	1285.11	804	77.86
MOEA/D	119225.18	31.79230	3.73503	1207.32	803	77.87
		31.79230	3.74222	1204.10	800	77.34
		31.88121	3.73017	1176.70	801	78.35
		31.79230	3.78271	1188.03	801	78.30
NSGA-III	114530.08	31.68357	3.71665	1244.65	808	83.64
		31.69567	3.71726	1279.17	809	86.12
		30.54910	3.71637	1235.65	800	75.09

Table 1: The results of Multi Objective Optimization problem of camp locations for the South Sudan Conflict

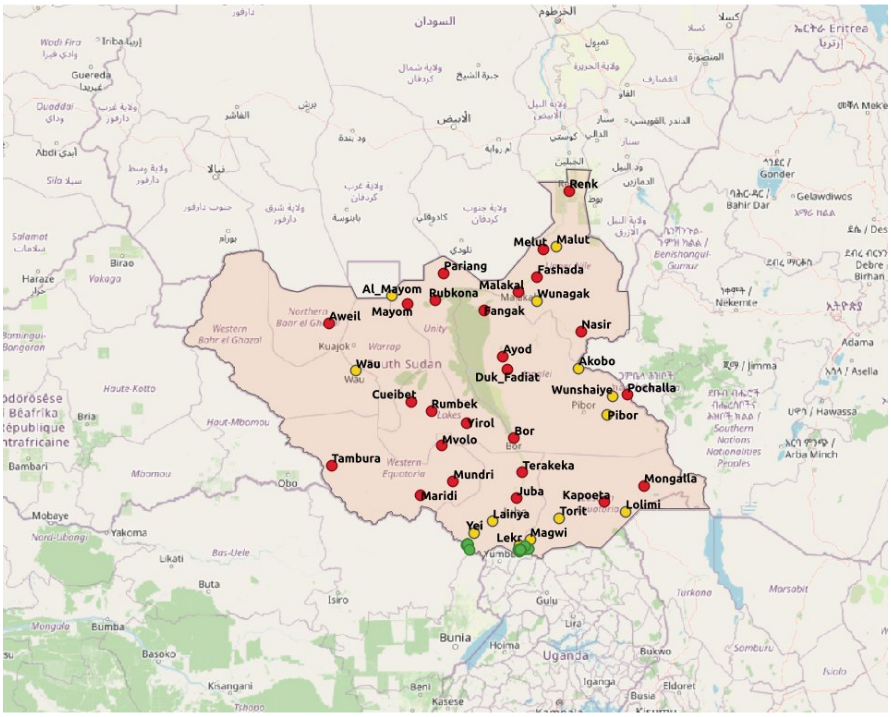


Figure 8: The optimized camp locations of the South Sudan conflict

3.2 Verification and validation status

At present, our main focus is the construction of new simulation instances, and we are in the process of gathering more accurate data from UNHCR for the new conflicts like Venezuela, Nigeria, Afghanistan, Syria, and Mali. Since the current validation data sets in the UNHCR portal are incomplete, we plan to do verification and validation phase when we gather enough data after reading and extracting reports. We also identify areas for automating data extraction and pre-processing. For instance, for location graph extraction and route calculations, we have started to use QGIS [6] and its useful plugins like ORS tools which help us have better location graphs and more accurate route calculations between all locations.

Data Coupling

To implement data coupling, especially for multiscale simulations, we explored the agent-based modelling assumptions and used two different coupling approaches, File I/O and MUSCLE3 [7]. We investigated possible effects of such coupling approaches on forced population displacement in South Sudan.

File coupling

In the Migration Multiscale simulation, both macro and micro models have coupling and ghost locations co-exist in both models where the agents are spawned from micro to macro through them. Ghost locations are the micro model conflict locations that we assume them residing in the macro model without having any link to other macro locations. Any agents arriving at coupled locations in the micro model will be passed on to the macro model using the coupling interface and vice versa. Whereas all agents in ghost locations will be spawned only to the micro model. All incoming and outgoing agents between macro and micro models are written in 'in.t.csv' and 'out.t.csv' where t represents the time step when coupling happens.

Model coupling using Muscle3

MUSCLE 3 is the third version of the Multiscale Coupling Library and Environment, and the successor to MUSCLE 2. Its purpose is to make creating coupled multiscale simulations easy, and to enable efficient Uncertainty Quantification of such models using advanced semi-intrusive algorithms from a non-distributed but parallel mode in a single (short) Python file to a distributed mode where the manager and compute element instances are started on multiple nodes in a cluster.

Weather Data Coupling

To address the different climate condition of the north east of South Sudan and investigate its effect on migration movement, we divided the South Sudan into Macro and Micro models and coupled the micro model with weather data provided by ECMWF including Precipitation Level data. This data set comprised 40 years precipitation data for South Sudan to identify the precipitation range (min and max). It also includes River Discharge level data when the rivers in South Sudan, in this case The White Nile River, shrinks or expands. ECMWF provided a daily river discharge data for South Sudan in the specified simulation period (river=1, no river=0) and river discharge return values for 40 years period (2 not extreme, 5 extreme and 20 very extreme).

3.3 Development of scope and pilot uptake

We regularly liaise with NGOs. E.g., IOM [8] on ruleset update, UNOCHA [9] on ethical aspects, UNHCR on new data collection, Co-design with Save the Children to address new research questions, like simulating IDPs. We also support new types of questions/scenarios (e.g., longer-term migration, migration across continents) for the ITFLOWS project [10]. We also use our models more widely as a tool to simulate goods movement for the STAMINA project [11].

3.4 Scaling and performance properties of the pilot use cases

The scalability of the Flee code has been extensively analysed, and is reported on in the WP3 deliverables. The core code scales to up to 16,384 cores for large problems (i.e. with 100M agents, see D3.4), and we also have shown that we can run ensembles consisting of 1,000s of runs using tools such as FabSim3 and QCG-Pilotjob [12]. In terms of production runs, most runs we present here are run in less than a day; using the Altair supercomputer [13] we have been able to rapidly do the requisite runs for migration forecasting reports for Save The Children.

3.5 Scientific Accomplishments

The work on Flee resulted in a range of scientific accomplishments. In addition to the computational papers discussed in WP3 deliverables, we published a paper on the Flee ruleset development in Philosophical Transactions of the Royal Society [12]. We also published a new location graph extraction algorithm in Nature Scientific Reports [14], and published a paper in the ICCS conference on the coupled South Sudan simulations [15]. In addition, there are several papers in preparation. These include, but are certainly not limited to, a journal paper about the coupled South Sudan simulation and a journal paper about applying many-objective optimization algorithms to determine the best locations for new camps in a forced migration simulation.

Perhaps at least as important as the papers are the collaborative research efforts that we have established. We are doing extensive research on the Tigray region in collaboration with Save The Children, and as of November 2021 the Institute of Migration has also become involved with this strand of research. We have already prepared collaborative research proposals on this topic, of which one was funded (the UKRI-funded SEAVEA project, which runs from August 2021 until 2024).

In addition, we have a close collaboration with Columbia University, which ranges from collaborative work on the Tigray simulations, to the integration of Flee with the US-funded

World Modellers project platform. Flee is now also a prominent application in the EU-funded ITFLOWS project (https://itflows.eu, 2020-2023), and is used as part of the EU-funded STAMINA project (https://www.stamina-project.eu, 2020-2023).

4 Status of the Covid-19 simulation pilot

The COVID-19 simulation pilot is distinct from all other pilots in one crucial way: it was not featured in the original HiDALGO proposal (although there was a "future application" task that allowed this activity to be within scope), and the main solver itself did not actually exist before March 2020, or M16 of the project's lifetime.

Through a dedication of substantial voluntary resources by the Brunel Department of Computer Science, along with collaborators from several London-based hospitals, and an essential effort and expertise contribution from the HiDALGO consortium, we have been able to establish a mature modelling code which can be used to forecast the spread of COVID-19 in local areas.

This code, the Flu And Coronavirus Simulator (or FACS, https://facs.readthedocs.io) is still very much being extended, and the status description of this particular pilot should not be seen as 'final'. Indeed, we foresee further progress between the submission date of this deliverable and the date of the final review, particularly in the area of code parallelization and scalability.

4.1 Documentation of the pilot implementation

The main component of this pilot is FACS, which is an agent-based simulation code that models infectious disease spread in a local context. FACS uses a susceptible-exposed-infectious-recovered-dead type of model (SEIRD) and it resolves each household and location explicitly by extracting relevant data from OpenStreetMap (among other data sources).FACS is able to resolve transmission within households, in public transport, as well as in a wide range of locations such as schools, hospitals, supermarkets, shops, leisure locations and offices. The code is also able to support a wide range of mitigation strategies, such as mask adoption, location closure, social distancing or household quarantine. There is also support for vaccinations and run-time changes in the characteristics of the disease. We use that latter functionality primarily to incorporate the spread of new variants. Lastly, immunity is modelled as a non-permanent state, allowing users to use FACS to model multiple waves of the pandemic.

This code is developed as part of the HiDALGO project, and is written in Python3 and distributed under a permissive BSD 3-clause license. We maintain a detailed documentation of FACS and its auxiliary components on https://facs.readthedocs.io. This website features a range of introductory explanations and tutorials, and has been demonstrated to be fit for the purpose of ensuring basic uptake by 3rd party users (e.g., within the context of the EU-funded STAMINA project).

FACS was initially implemented as a rapid prototype agent-based simulation, and was iteratively refined after having been validated and applied for local COVID-19 forecasting reports. To streamline the simulation development, particularly for the purpose of using the code for different locations, we established a Simulation Development Approach [16] as well as a range of tools for preparing new location graphs, and automating other essential components of the simulation, and the simulation development, workflow.

The essential repositories include:

https://www.github.com/djgroen/facs - main code

https://www.github.com/djgroen/FabCOVID19 - FabSim3 automation module to enable easier scenario analyses and post-processing.

https://www.github.com/djgroen/covid19-preprocess - tools to support pre-processing activities.

We used to have a (private) post-processing repository for validation and analysis, but this repository is now gradually being phased out. Instead, these functionalities will be offered to the public through the FabCOVID19 plugin.

As of Q2 2021, we are developing a parallel version of FACS (available in the 'pfacs' GitHub branch). This version currently runs and produces results, and we are currently in the phase of ensuring that this parallel version produces results that are in line with sequential executions of the code (minor mismatches still exist at the time of writing).

4.2 Verification and validation status

Verification and validation is very much a work in progress for FACS, although we have performed validation for several boroughs in London (see e.g. [16] and various non-public NHS forecasting reports).

In terms of verification, we currently maintain a narrow suite of functional tests for FACS. This suite was deliberately kept narrow to allow easier modification of core code routines but is now gradually being expanded as of Q3 2021.

The parallel version is undergoing verification relative to the serial version.

4.3 Development of scope and pilot uptake

The initial scope of FACS was to model the spread of COVID-19 in a local area, with a local area being a region encompassing 100,000-300,000 households. The choice of scope was straightforward, as partner BUL received a request from a local NHS trust to develop a simulation for exactly that context. We slightly expanded the scope later in 2020, such that the model could also be used to help predict hospital bed occupancy rates. Here, FACS loosely couples to a hospital simulator named CHARM, which is developed in the context of the STAMINA project (https://www.stamina-project.eu).

As of today, the plan is to enable FACS to simulate larger regions, primarily by parallelizing the code. In addition, we are generalizing the disease specification mechanisms, such that FACS could also be used for other air-transmitted diseases, such as Influenza.

Lastly, FACS can be used to generate synthetic footfall data for various locations, such as supermarkets and hospitals. Although we have applied this functionality in a few informal collaborations, we would like to make further development progress on these routines before advertising this functionality more widely.

In terms of uptake, FACS was initially used to inform strategic decision-making for two NHS trusts, namely the London North West Hospitals Trust and the Hillingdon Hospitals Trust. In addition, it has been applied by a group of collaborators in Pakistan to model the spread of COVID-19 in Islamabad. Within HiDALGO, we are investigating the application of FACS to model disease spread in Madrid and within the STAMINA project we are intending to use FACS for a range of trials. These trials take place respectively in the UK (nationwide, contingent on parallelization developments in HiDALGO), Lithuania, Romania and Turkey. Lastly, we have an informal collaboration with researchers at the University of South Florida, and are looking into applying the code to model one or more US cities as well.

4.4 Scaling and performance properties of the pilot use cases

Sequential FACS runs typically take 1 to 8 hours on a local machine. The parallel version is still being tested, and we aim to provide first parallel scalability results by the time of the final review. Our aim is to have the code scale to at least 128 cores per run by the end of the project.

4.5 Scientific Accomplishments

The main accomplishment in this pilot is in our delivery of 13 forecasting reports to two NHS Trusts in West London, as well as in establishing a European code that can directly incorporate mapping data to construct highly localized digital simulations of COVID-19 spread in local areas.

FACS was also recognized as a highly important code by the STAMINA consortium, which subsequently chose to incorporate the model in several of its trials, even though the code did not yet exist when the STAMINA Grant Agreement was first established.

The work performed on FACS was peer-reviewed and resulted primarily in one journal publication [16], as well as a range of invited talks (see D6.5 [17]). Several additional publications are currently in preparation.

5 Status of urban air pollution pilot

Within the urban air pollution pilot we have made progress in many areas: most notably we have validated our simulations, established a reduced order model, and updated our workflow to include more scalable and open software components.

5.1 Documentation of the pilot implementation

The overall workflow of the Urban Air Pollution application currently looks as follows:

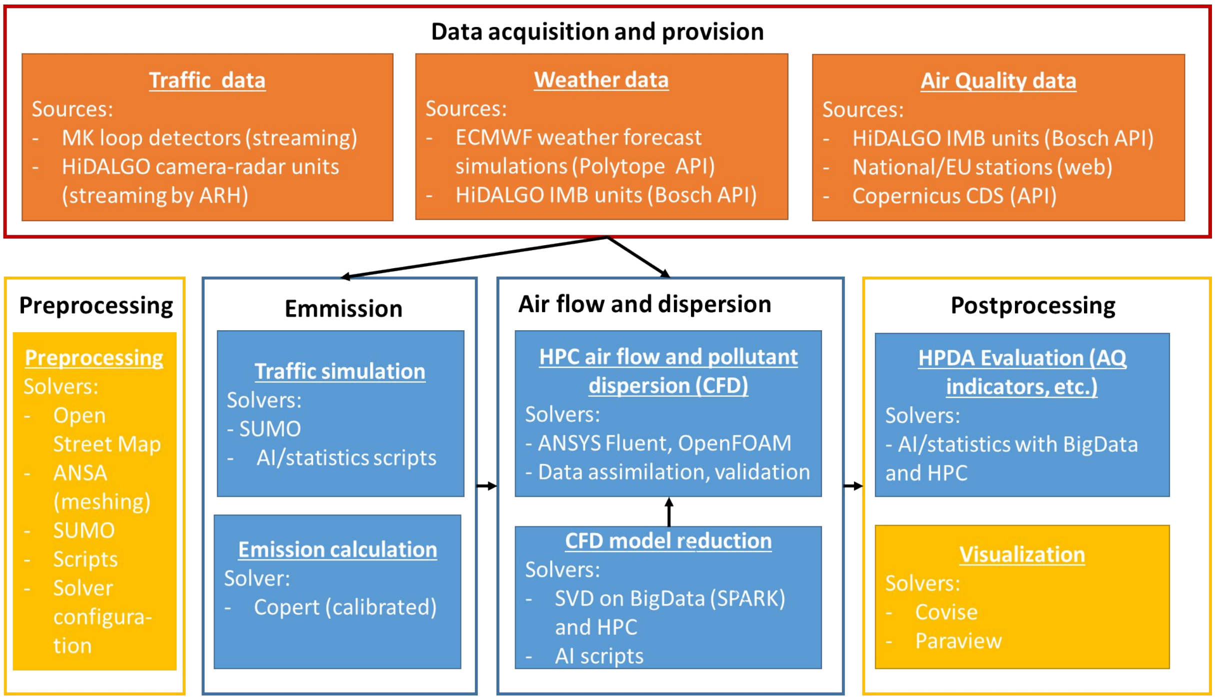


Figure 9: Urban Air Pollution workflow.

We have defined this workflow in a TOSCA blueprint, which in turn can be used by Cloudify for execution, and relies on CKAN for the various data staging functionalities. We also provide a web interface which has been implemented in Python Django. As part of this interactive portal we also developed a visualisation tool and an air quality index evaluation component. Detailed information on these functionalities can be found in the WP5 deliverables.

Model order reduction

An instance of the coupled air pollution application typically requires an ensemble of runs. To reduce the computational cost of these examples we have developed reduced order models. These models provide an accuracy that is slightly inferior to the full-order models, but at a computational cost that is considerably cheaper than the full-order models.

We have developed a reduced order model using the Proper Orthogonal Decomposition method. This model relies on an offline training stage, followed by an online prediction stage. We present a diagrammatic overview of both stages below:

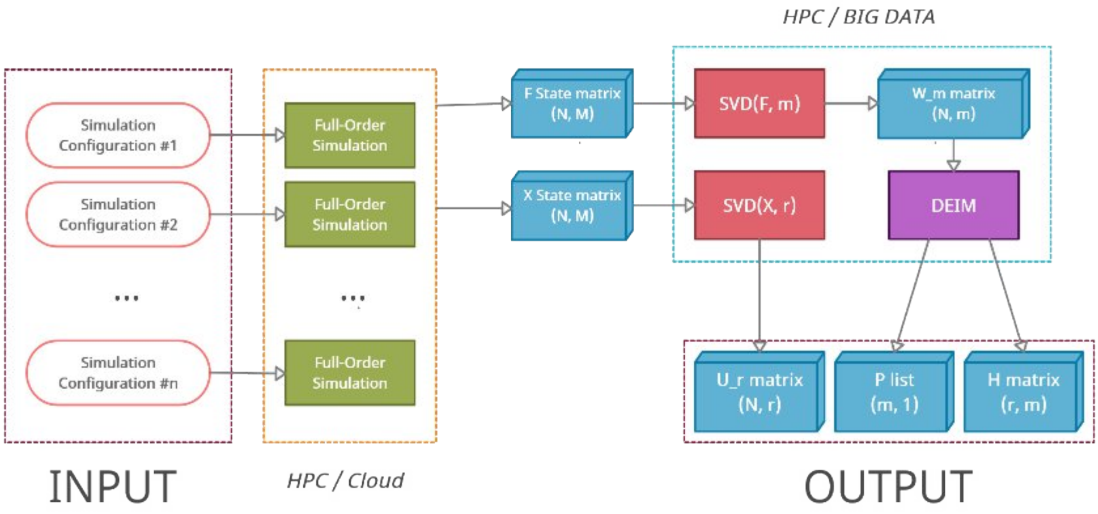


Figure 10: Overview of the offline training stage of the reduced order model.

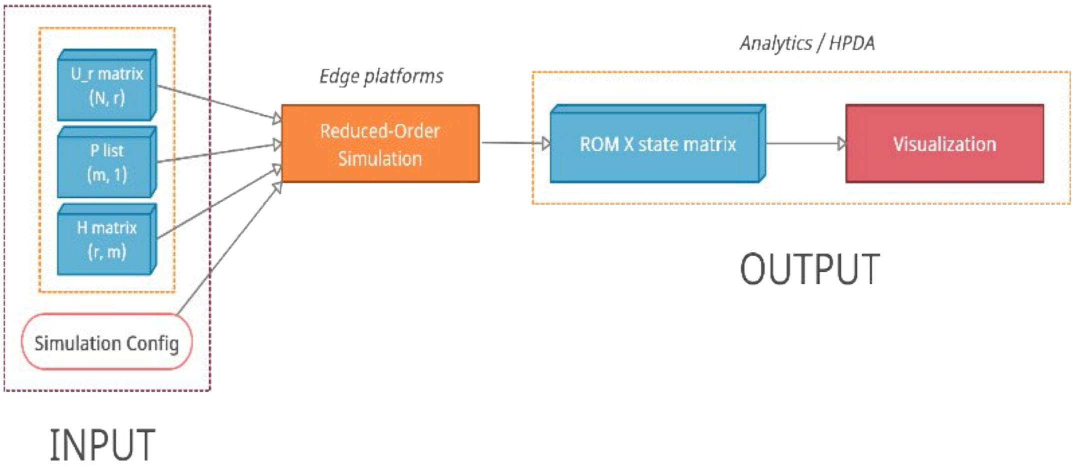


Figure 11: Overview of the online prediction stage of the reduced order model.

We tested the Reduced Order Model (ROM) in a range of different settings. As an illustrative example we present a validation against the full model for the city of Antwerp:

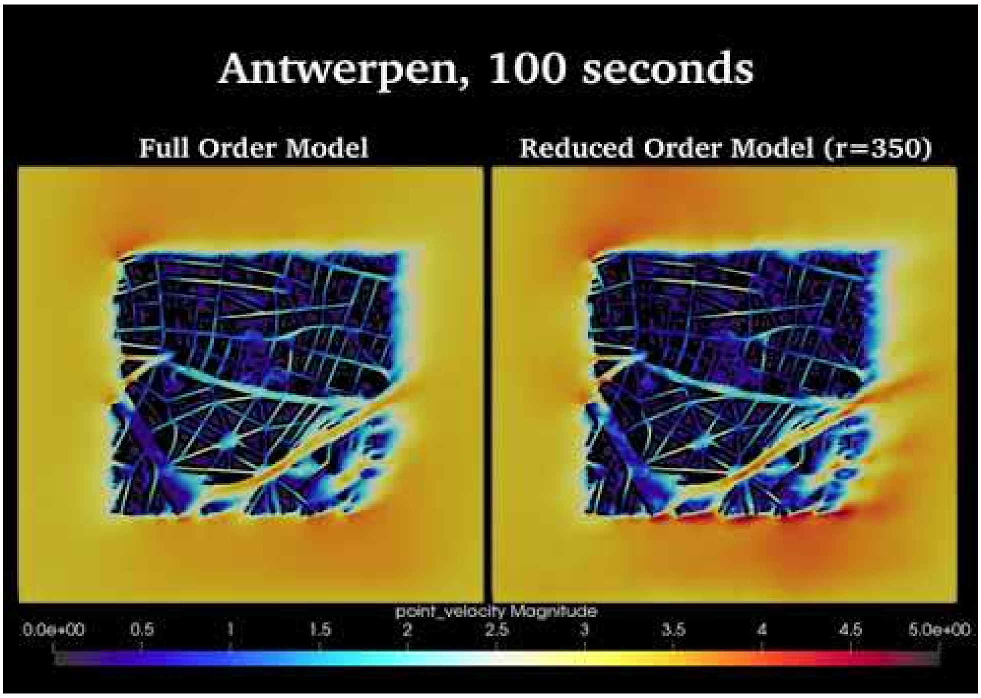


Figure 12: Comparison of the reduced order model prediction with the full order model prediction for the city of Antwerp.

Here we can observe that the ROM reproduces many of the essential airflow characteristics in the city. In terms of performance, the ROM is approximately 30 to 100 times faster than the full scale model for typical cases. If the user spends particular time for training, the performance gain is substantially situations. In particular, SZE trained a model from 12 runs of the full order model and used the resulting reduced order model to Antwerp test case to simulate one full day of 6th May 2016 (data of the 1st stage of the Intercomparison Exercise), which resulted Figure 12 below.

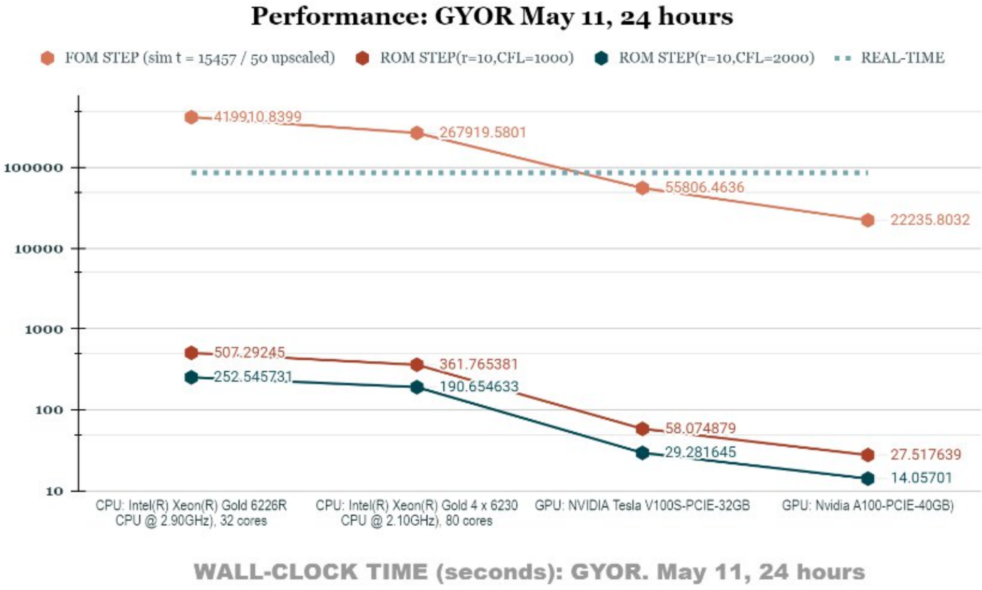
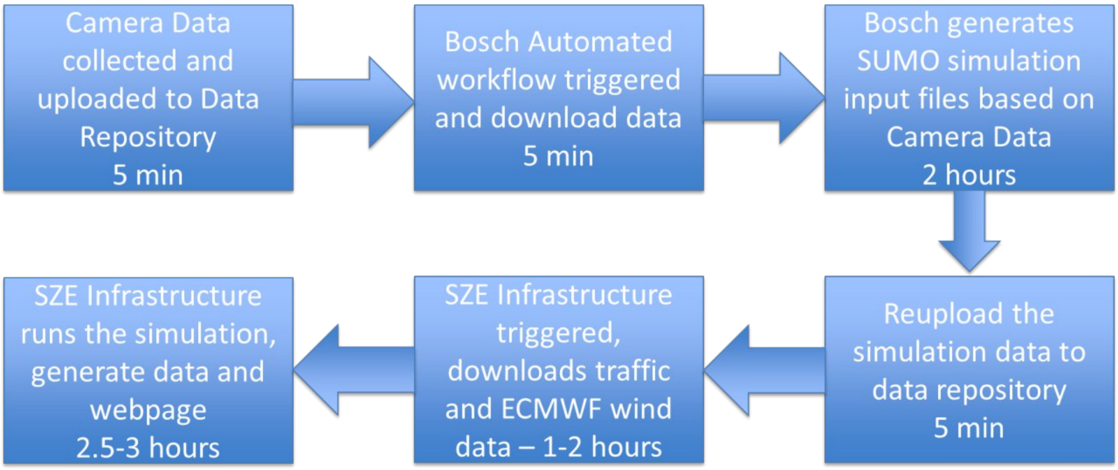


Figure 13: Comparison of computation time (in seconds) for the air flow simulation of the city of Gyor for 1 day physical time. Observe that reduced order simulation of 1 day needs 14 seconds on an NVIDIA A100 only.

Automated reanalysis

As part of this pilot, we also have managed to establish daily automated runs of Gyor. These runs contain a coupling to ECMWF weather prediction and traffic modelling, and are run on a daily basis at 00:15AM. The workflow performed during these runs is as follows:



Camera Data collected and uploaded to Data Repository 5 min

Bosch Automated workflow triggered and download data 5 min

Bosch generates SUMO simulation input files based on Camera Data 2 hours

Reupload the simulation data to data repository 5 min

SZE Infrastructure triggered, downloads traffic and ECMWF wind data – 1-2 hours

SZE Infrastructure runs the simulation, generate data and webpage 2.5-3 hours

The simulation results are publicly available, and are shared at: https://hidalgo-project.eu/use-cases/pollution/AQ-reanalysis-Gyor. Results contain, for each day, a set of graphs of the simulation results: the first image is a link to an interactive 3D visualizer, in which urban wind and pollution concentration can be seen, and the second image routes the user to a time series of graphs of the pollutant concentration at 5 meter height (2D slice). Also, the graph of the time-series of the total emission is provided.

5.2 Verification and validation status

The UAP pilot has been applied to a wide range of cities, including Gyor, Stuttgart, Graz and Antwerp. We have specifically performed validation studies of the Urban Air Pollution application for the cities of Gyor, Antwerp. The evaluation of the validation simulations to Győr for a full year is under processing. Here below, on Figure 14 and Figure 15 we present two validation graphs for the results for the 1st stage intercomparison exercise of FAIRMODE, which is the direct comparison of NOx simulation and measurement results at the street side station and background station, respectively. In the validation experiment the emission is given from statistical top-down data and might be crude approximation of one actual real scenario.

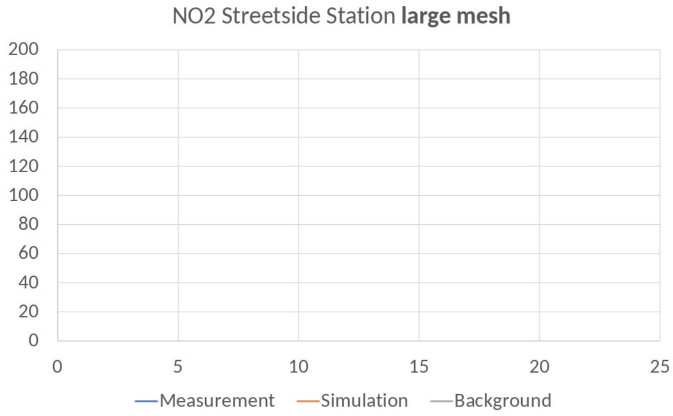


Figure 16: validation of the UAP pilot in Antwerp for a street side station.

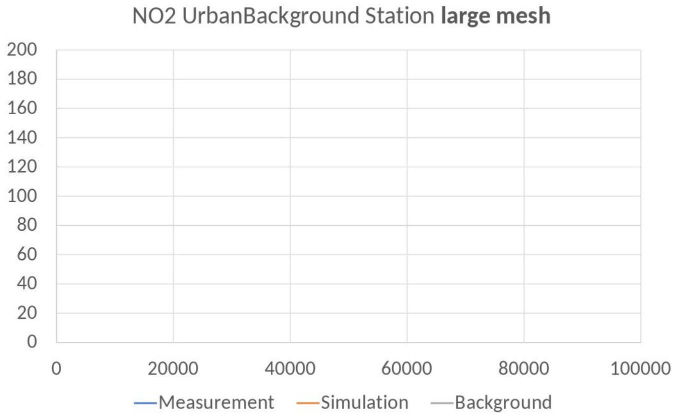


Figure 17: validation of the UAP pilot in Antwerp for a station with an urban background.

The mean average percentage error of these two cases were 20.0% and 27.9%, respectively, which are acceptable in the considered uncertain situation. The overall behaviour of the approximation is much below the final average error, which is due to the discrepancies during the hump at sunset (on the right edge of both plots) when the planetary boundary layer drops rapidly having a strong impact on the mixing coefficients. The discrepancy could be cured with a more detailed meteorological model than the global ECMWF model, which is out of scope of the present project.

5.3 Development of scope and pilot uptake

The uptake of the urban air pollution pilot has rapidly expanded, as described in the earlier sections. We established a close collaboration with Bosch, and have applied the simulation to five different cities, collaborating with local municipalities in this context. In addition, SZE is participating in EC's FAIRMODE CT4 (Forum for Air Quality Modelling in Europe, see [16]), which evaluates high resolution modelling techniques according to the EC Air Quality Directive (see [17]). CT4 has been running an Intercomparison Exercise for verified data for Antwerp and SZE participates with UAP – SZE is the only partner so far from Europe that runs an unsteady microscale simulation for a full year.

5.4 Scaling and performance properties of the pilot use cases

Over the course of HiDALGO, we have modified the setup of our pilot such that it becomes more scalable and relies on open software where possible. In this work we applied some joint work for parallel code optimization with EXCELLERAT CoE. Most notably we now rely on the OpenFOAM solver for many of the airflow calculations: this solver has now been benchmarked as part of the work in WP3. In addition to the graphs delivered in line with WP3, in this document we present the following representing figures from the results on the HAWK cluster.

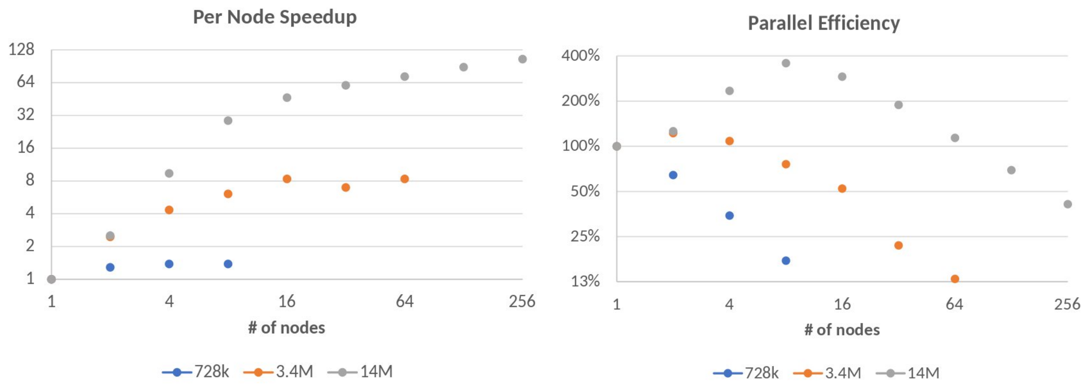


Figure 18: Per node speedup of UAP OpenFOAM module on Hawk – scalability up to 32768 cores for Győr.

The current best run for single simulation is with a 14M size Győr mesh with 1 meter resolution. The largest number of cores used is 32768 on 256 nodes on the cluster HAWK, of which estimated computer performance counts on 1024 TFLOPS. We observed superlinear speedup up to 64 nodes and above 50% efficiency up to cca. 200 nodes for the largest problem, which seems reasonable.

As larger mesh sizes are still manageable, an additional 67M size mesh of the city Paris with 3-meter ground level resolution is in prepared and benchmarking is ongoing. This result will be compared with those renowned experiments of [18]. According to our estimate, this single simulation will use more than 100.000 cores with strongly increasing speedup.

5.5 Scientific Accomplishments

Although our focus has been primarily on widening the uptake and establishing direct collaborations with external stakeholders, we did publish a number of papers already. One example is our recent publication for the International Conference on High Performance Computing [18], which presents our workflow as a whole. The results of the project on the UAP were presented in several highly acknowledged scientific conferences, both computational (e.g. at NUMDIFF-16, see https://sim.mathematik.uni-halle.de/numdiff/Numdiff16/) and more environment oriented ones (e.g. as invited lecture at HARMO20, see https://harmo20.ut.ee/) and received positive feedbacks, in particular the achieved unexpectedly large stability property of the reduced order model (explicit Euler time-stepping is stable for CFL-number up to 2000 for POD in case of the Antwerp use case though for classical solver this threshold is below CFL=1). The publication of the corresponding materials are under preparation.

6 Status of social network pilot

6.1 Documentation of the pilot implementation

In this section, we describe the recent advancements of various aspects of our pilot. We focus on the simulator itself (available at https://github.com/sarming/propagation) and on the eigenvalue histogram calculator (available at https://github.com/sarming/kpm), which is part of the validation module. We report newly introduced functionality and give a short overview of some recently performed performance optimizations.

Furthermore, we developed a new approach for synthetic graph generation. The new approach models directed graphs based on crawled follower networks from Twitter, whereas the former approach focused on undirected graphs.

Refining the Simulation Model with additional Parameters

In order to allow our simulation to more closely replicate the message dissemination patterns of real world social networks, we introduced two additional parameters. Values for these parameters need to be specified for each so-called *feature vector* (also called feature class) of tweets that should be simulated. In short, a feature vector is assigned to each tweet before it is simulated to describe its characteristics. For example, one such simple feature vector could describe tweets that contain a link and are initiated from a verified twitter author. A more detailed description of these feature vectors was given in Section 6.1.2.2 of Deliverable D4.3 [1].

In order to better follow the idea behind our parameters, consider the following quick sketch of the behaviour of our simulator when simulating the spread of some tweet *t*. Each time tweet *t* arrives at some new user *u* – either because it was retweeted by *u* or because it originates from *u* – every neighbour of *u* will retweet *t* with a fixed probability *p*. This process repeats until *t* is no longer retweeted by any node. Our new parameters refine this process as follows.

- discount_factor: The nodes retweeting some tweet *t* are structured in a tree. At level 0 we have the node at which *t* originates from. At level 1 we have the nodes that retweeted *t* from the author of the tweet. At level 2 we have those that retweeted *t* after learning about *t* from a level 1 node and so forth. The discount_factor is a constant in [0,1] which reduces the retweet probability *p* with every subsequent level. The idea behind this parameter is to capture the fact that it is unlikely for this dissemination tree to have too many levels. Overall, setting a value below 1 for this parameter reduces the number of retweets made in our simulation and thereby prunes the dissemination trees.

- correlation_probability: Extending our simulator above, we allow the neighbours of *u* to retweet *t* in an additional way. That is, if at least one neighbour of *u* retweets *t*, then every other neighbour of *u* has an additional attempt at retweeting *t* with some fixed probability. The exact probability for a successful attempt can be specified with this correlation_probability parameter. The intuition behind this parameter is that, if a tweet is retweeted at all, then it will most probably be retweeted multiple times. This parameter nicely contrasts the discount_factor as it increases the overall number of retweets made.

In Section 6.2 we examine the effect these parameters have on the accuracy of the simulator.

Parameter Learning Approaches

The current twitter simulation model relies on 3 important parameters which can be specified for each feature vector. These parameters are the propagation_probability as well as the more recently observed and introduced discount_factor and correlation_probability (see e.g. [19, 20, 21] on the propagation behaviour of messages in Twitter). The propagation_probability just corresponds to the value *p* which we used in our sketch of the simulator given in Section 6.1.1. That is, it denotes the probability that a neighbour of a node *u* decides to retweet a tweet of *u*. The other two parameters are described in detail in that section as well. The value of *p* can be computed directly from one of the input files. However, choosing values for the other two parameters is not straightforward and we employed three approaches to find appropriate values for a given data set.

Currently, all three of these approaches aim to minimize the error in the average number of retweets when compared to the ground-truth data. Our application requires the usual two input files consisting of (i) an input graph describing the network structure in which the simulations should be performed, and (ii) the statistics table described in Section 6.1.2.2 of Deliverable D 4.3, which in essence contains a list of tweets along with their assigned feature vector and how often they were retweeted. This second file in (ii) is used to compute the average number of retweets per feature vector, which we consider to be the ground-truth.

- **Binary Search:** In this approach, values for discount_factor and correlation_probability cannot be learned at the same time. One of the two parameters needs to be fixed beforehand – either by using previously computed values or default values. The other parameter is optimized via a binary search over [0, 1]. Such a binary search is performed for each feature vector and the ground-truth data is consulted to decide in which direction the search needs to continue. Note that this approach is only possible because both above parameters cause the expected number of retweets to either monotonically increase or decrease.
- **Grid Search:** In this case, values for both discount_factor and correlation_probability are learned simultaneously. We separate the 2-dimensional search-space into a grid

and perform a simulation run at each data point in the grid. We select those parameter values, which most closely reflect the average number of mean retweets of the ground-truth data. This search also needs to be performed for every existing feature vector. As multiple points in the grid can be searched independently, this approach lends itself to massive parallelism. In fact, as reported in detail in Deliverable D3.5 [2], we performed this search on up to 100K cores in parallel. In the same deliverable, we also give additional details on this variant for parameter optimization.

- **Bayesian Optimization (Kriging):** In this approach we use a Gaussian process (GP) to model our simulation – specifically, the function from the parameters discount_factor and correlation_probability to the mean retweets the simulation returns under those parameters. This allows us to iteratively use the GP to find a promising parameter point (or points), run the simulation on this point and then subsequently update the GP with the result. What constitutes a promising point – an instance of the exploration/exploitation trade-off – be expressed by a so-called acquisition function to be maximized. In our preliminary experiments we maximize the following instance of an Upped Confidence Bound (UCB): $f(x) = \kappa\,\sigma(x) - |\mu(x)-y(x)|$ where x is the parameter point, y is the ground truth, $(\mu,)$ is the GP, and κ=2.5 is a freely chosen weight. As with the other approaches this search is done for each feature vector independently and in parallel.

In Section 6.2, we report some results that describe the effect that optimized parameters have on the accuracy of our simulator.

Performance Improvements and Optimizations

Quite some efforts in the previous weeks and months were spent to improve the performance of our pilot application. We made various performance improvements to, both, the simulator and also the eigenvalue histogram calculator. As these changes are described in detail in Deliverable D3.5 [2] we only give a quick overview.

On the side of the social network simulator, we performed three optimizations:

- **Task reordering:** We apply heuristics to ensure that our simulation starts with the simulation of tweets which are likely to be frequently retweeted and thus take longer time. This allows balancing the workload among MPI workers more evenly.
- **Task splitting:** We introduced a parameter which gives more control of the batch size of tweets that are simulated. Again, this improves the workload distribution.
- **Numba JIT:** We use the Just-In-Time compiler (JIT) of the numba [22] python package to convert one of the hotspots of our simulation into efficient machine code.

On the side of the eigenvalue calculator, we made a single improvement that yielded significant performance gain on processors with many cores.

- **ccNUMA optimization:** Storing the input graph multiple times in the shared memory of each node in order to avoid contention in memory access.

In Section 6.4 we explore the impact these improvements have on the performance of our applications.

New Model for Synthetic Graph Generation

In Y3 of the project, we developed a new approach for synthetic graph generation. The new approach models directed graphs based on crawled follower networks from Twitter, which include users that tweet, retweet, quote, or reply on pre-defined topics.

We analyse and compare the structure of the crawled and the created graphs, and simulate certain algorithms for information dissemination and epidemic spreading. The results show that the created graphs exhibit very similar topological and algorithmic properties as the crawled real-world graphs, providing evidence that they can be used as surrogates in social network analysis. Furthermore, our graph model is highly scalable, which enables us to create graphs of arbitrary size with (almost) the same properties as the corresponding real-world networks.

The synthetic graph generation works as follows. We extract information from the crawled network graphs and fit χ^2 distributions with different parameters to the reciprocal, in- and out-degrees of the nodes from which the node degrees for the synthetic graphs can be sampled. Since there is a high correlation (Spearman's rank correlation coefficient ρ) between the degrees in the crawled graphs, randomly sampling the node degrees results in unrealistic joint degrees for nodes, such that we sample data from a normal distribution with mean vector $m = (0,0,0)$ and covariance matrix according to the correlation between the degrees. The sampled data is then transformed into node degrees R,I,O (reciprocal, in, out) in a two-step procedure [23] that have approximately the same rank correlation as the node degrees of the crawled graphs. To establish connections between nodes, we connect them via reciprocal or directed edges while avoiding self-loops (node connected to itself) and parallel edges (two edges in the same direction between two nodes).

At first, we compute the reciprocal connection probability $P(i-j)$ between all node pairs (i,j) with $1 \leq i,j \leq n$ according to their sampled reciprocal degrees $R(i)$ and $R(j)$ and the self-loop probabilities $P(i-i)$. As we want to generate a simple graph, the self-loop probabilities are uniformly distributed over all possible reciprocal edges via addition. A reciprocal edge is then generated with the computed probability.

We are using the same approach for directed edges, where the sampled out-degree $O(i)$ and the sampled in-degree $I(j)$ of each node pair (i,j) is used to compute the directed connection probability $P(i \rightarrow j)$ and the probability for directed self-loops $P(i \rightarrow i)$ is uniformly distributed over all possible directed edges via addition. We avoid parallel edges in the created graph, such that the probability of a directed edge that has already been sampled as a reciprocal edge is also uniformly distributed over all possible directed edges via addition. Directed edges are eventually generated with the computed probabilities.

The created graphs have approximately the same degree distribution and the same density as the crawled graphs, but the average clustering coefficient (number of actual triangles over number of potential triangles) is much lower. Therefore, we utilize the technique of edge rewiring while preserving the node degrees by swapping two edges at a time in the graph to increase the clustering coefficient [24]. For each node x in the lower 95-percentile of node degrees, we choose a fraction of first degree neighbour pairs (y_1,y_2), depending on the degree of x and check if they are connected. If they are not connected, we sample a neighbor z_1 of $y_1$and a neighbor z_2 of $y_2$with z_1 and z_2 not being connected and try to swap the edges between the node pairs (y_1,z_1) and (y_2,z_2) for edges between the node pairs (y_1,y_2) and (z_1,z_2), while preserving the node degrees. A successful rewiring adds a triangle for node x and increases its clustering coefficient.

6.2 Verification and validation status

Validation of the Average Number of Retweets

We pick up on the discussion started in Deliverable D 4.3, where we briefly reported first results concerning the accuracy of our simulator. Since then, the introduction of the new parameters described in Section 6.1.1 as well as the implementation of learning algorithms for these parameters significantly increased the accuracy of our simulator.

One metric we can use to compare the accuracy of our simulator to the behaviour of real world social networks is the average number times a tweet is retweeted. For our tests, we considered neos and fpoe, two of the twitter data-sets, as an input for our simulator. Both this data-sets consist of tweets and twitter users that discuss the respective political parties NEOS and FPÖ. The data was acquired around the 2019 Austrian national council elections. For each data-set we performed 4 experiments.

- **Binary Search:** We used the binary search approach described in Section 6.2 to learn the parameters. More precisely, we first learn values for the discount factor while using the default value 1 for correlation_probability and then we learn the correlation_probability while using the default value 0 for the discount_factor.
- **Grid Search:** We apply the grid-search algorithm as described in Section 6.2. We used a grid consisting of 40x40 data points for each feature vector
- **Bayesian:** We apply Kriging as described in Section 6.2. using UCB and 100 iterations.

- **Default:** We use default values for both correlation_probability and discount_factor (0 and 1, respectively). This leads to the same behaviour as before the introduction of these two parameters.

In each experiment we report the so-called Mean Average Percentage Error (MAPE) in the number of retweets when compared to the ground-truth data. The MAPE is calculated for any of the four experiments as follows. First, we perform a simulation run that uses the learned parameter values. This yields the average number of retweets a_f for every feature vector *f*. Additionally, we know the ground-truth has average retweets r_f. The MAPE is then computed by summing the error $|a_f - r_f|/r_f$ over all feature vectors *f*. The resulting error values are illustrated in Figure 19.

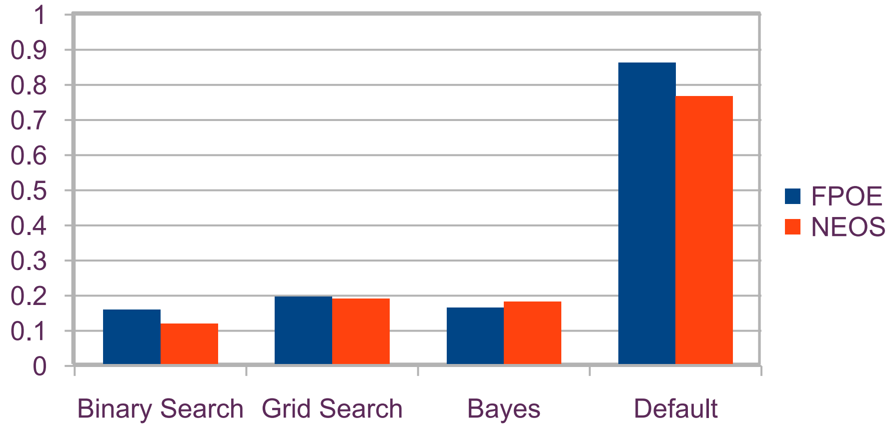


Figure 19: MAPE of our simulator for the neos and fpoe data-sets when using parameter learning beforehand

As we can see, all of our learning approaches significantly outperform the default case in which no parameter learning is used at all. While the error lies well above 80% in case of the default setting and fpoe data-set, it is now at most 17% for both data-sets when using the binary search optimization strategy.

Exploring additional Validation Metrics

As briefly described in Section 6.1.1, for every tweet *t*, one describe the set of users that receive and retweet this tweet with the help of a tree. More precisely, each node inside this tree corresponds to a user that either is the origin of the message or retweeted the message. An edge *(u,v)* is inserted if the user corresponding to node *u* retweeted the message after learning it of *v*.

The general idea is to compare the structure of these tweet trees that are generated by our simulation to those trees that can be observed in the real world. As a first step, we plan to start by comparing the number of nodes which lie on a certain level of these trees.

To support this approach, we added functionality to our simulator that allows the output of these dissemination trees. Additionally, we developed a HPDA method which allows for the calculation of these statistics out of the real world data we previously obtained of twitter. This method is explained in more detail in Section 6.3 of Deliverable 3.5. At this time, the validation process with the help of these trees is still in progress.

Validation of the synthetic graph generation

To evaluate the graph generation method, our goal was to create graphs with the same number of nodes, a comparable density, and a similar average clustering coefficient (CC) in the largest weakly connected component (LWCC) as the crawled graphs. Table 2 contains numbers for topological features in graphs, which indicate that our method generates graphs that are similar to the crawled ones.

Graph parameters	G_1	G_2	G_3	G_4	G_5
Nodes	11,015	21,291	50,133	459	3,580
Edges	377,457	2,570,452	4,832,226	5,435	54,735
	381,627	2,527,541	5,049,608	5,499	57,249
Density	0.0031	0.0057	0.0019	0.0259	0.0043
	0.0031	0.0056	0.0020	0.0262	0.0045
LWCC	10,931	21,281	49,999	452	3,570
	10,166	21,148	48,607	446	3,354
Average CC	0.201	0.285	0.198	0.354	0.277
	0.228	0.291	0.222	0.300	0.187

Table 2: Topological features of the crawled graphs (line 1) and the created graphs (line2).

Concerning algorithmic properties, we compared the behaviour of information dissemination with the push-pull model [25] and the so-called Susceptible-Infected-Recovered (SIR) model [26].

In our experiments (Table 3), we simulated the push-pull model on the LWCC of the crawled and created graphs. At the beginning, we assigned a message to $2 \cdot \log n$ nodes, where n is the number of nodes in the graph. Then, the process takes several rounds, where in each round every node selects a neighbour and transmit the message to the neighbour if it is present at the node. The process runs until all nodes received the information. Each simulation run is repeated 100 times for G_1, G_4, G_5 and 10 times for G_2 and G_3.

	G_1	G_2	G_3	G_4	G_5
Rounds	13.92	10.64	17.24	9.79	12.40
	10.78	8.95	11.22	6.30	7.62

Table 3: Average number of rounds for message dissemination in the push-pull model

In the simplest variant of the SIR model, we assume that at the beginning a small number of nodes is infected. In each round, any infected node spreads the infection to each of its outgoing neighbours with some probability, independently. Then, the nodes that spread the infection in this round become recovered and cannot become infected again. For a better comparison, we consider the fraction of recovered nodes after 100 rounds for G_1, G_4, G_5 and 10 rounds for G_2 and G_3. From the results given in Table 4 we observe that the difference is in most of the cases almost negligible, i.e., is less than 0.03.

Infection probability	G_1	G_2	G_3	G_4	G_5
p = 0.3	0.8104	0.9572	0.9274	0.7431	0.7526
	0.8097	0.9656	0.9109	0.8562	0.7930
p = 0.1	0.6573	0.8812	0.8423	0.5257	0.5317
	0.6500	0.8881	0.8098	0.6078	0.5872
p = 0.05	0.5203	0.7994	0.7548	0.3360	0.3755
	0.5302	0.8098	0.7248	0.3769	0.4327
p = 0.01	0.1470	0.5015	0.4427	0.0435	0.0370
	0.1930	0.5275	0.4560	0.0458	0.0318
p = 0.005	0.0105	0.3418	0.2807	0.0356	0.0080
	0.0052	0.3581	0.3063	0.0392	0.0084

Table 4: Fraction of recovered nodes for different probabilities p in the crawled graphs (line 1) and the created graphs (line 2)

6.3 Development of scope and pilot uptake

The main stakeholders for this pilot application are representatives of the mass media and policy makers. Concerning the mass media, we plan to address the press as soon as the final simulation runs are performed, and we are able to make predictions about the spread of messages in social media. This will be done within a sequence of press releases, in which we inform about our simulation framework and its advantages for the public. Furthermore, we will address selected members of the national audio-visual media in Austria, represented by the ORF. We expect that the sequence of press releases as well as the echo of our results in the media will attract the attention of a number of national policy makers.

6.4 Scaling and performance properties of the pilot use cases

In this section, we present an overview of the scalability properties of our social network simulator (SN-Simulator) as well as the eigenvalue calculator (KPM). At the same time, we also illustrate the effect of optimizations described in Section 6.1 by comparing our applications to previous versions. In Section 3.3 of Deliverable D 3.5, we give a more detailed description of theses benchmarks, including a list of all employed parameters and additional results.

Performance Improvements of the SN-Simulator

In order to assess the quality of the improvements described in Section 6.1.3, we compare version v0.3 of the SN-Simulator to the previous version v0.2. We use the fpoe and neos twitter data-sets as input (these data-sets were briefly introduced in Section 6.2.1). In all experiments, we use the same parameter values for both versions of our software. A more detailed analysis is provided in D3.5.

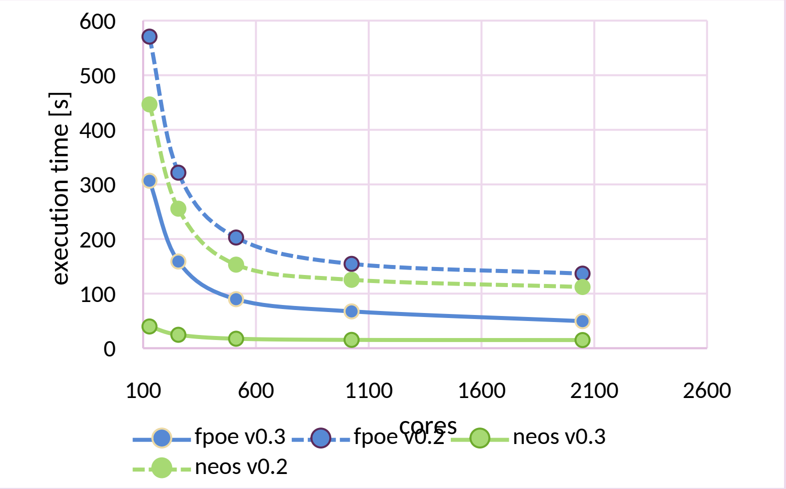


Figure 20: Execution times of the SN-Simulator when using the fpoe and neos data-sets when using an increasing amount of cores on Hawk

As we can see in Figure 20, the new version of our application significantly outperforms the previous version. For the neos data-set, on only 128 cores we can even observe an execution time that is 10 times shorter. Also in case of the fpoe data-set we can observe a speed-up of at least factor 2 in every data point.

Performance Improvements of the KPM

We compare the performance of the current version v0.2 to the previous version v0.1 on the Hawk supercomputer. To this end, we computed an approximation of the eigenvalue histogram of the friendship relationship graph of the pokec social network (~990K nodes, called pokec_trimmed on https://ckan.hidalgo-project.eu/dataset/pokec-relationship-graphs). The computed spectrum of this graph was illustrated in Figure 9 of Deliverable 4.3 [1].

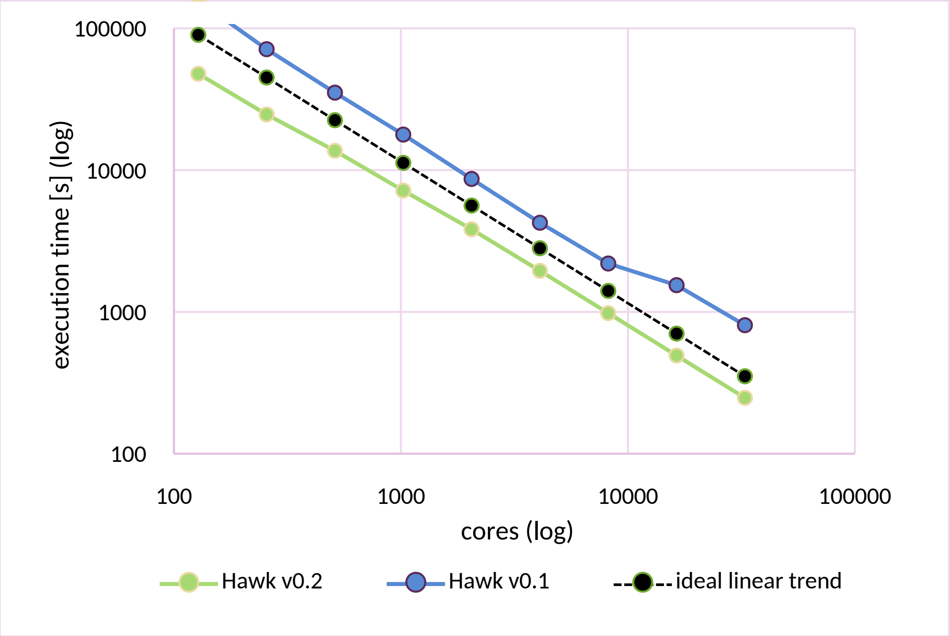


Figure 21: Execution time for the calculation of the eigenvalue histogram of the pokec graph on Hawk when using an increasing amount of cores

As we can see in Figure 21, our application maintains its strong scalability properties while also outperforming the previous implementation by a factor of at least 2. This is a result of the ccNUMA optimizations mentioned in Section 6.1.3.

6.5 Scientific Accomplishments

The scientific accomplishments within this pilot application are three-fold. First, we have designed new methods for analysing real-world networks by deriving a clustering procedure, which detects overlapping clusters in real-world networks. This is achieved on one hand side by clustering the neighbourhood of each node and then removing/merging clusters according to some optimization function. Another method uses so-called state dynamics, which colour the nodes of the graph, and according to the history of colours the node is assigned to one or more communities. The analysis of the corresponding state dynamics have already been published [27, 28], while their application to real-world networks will be described in a paper, which is currently under preparation.

The second scientific field we are working on in this pilot application is the generation of random graphs for sub-networks of Twitter on which information with regard to a certain topic is spread. The method has thoroughly been described in Section 6.1.4. A paper on this topic is currently under review.

The third scientific field is on the spread of information in Twitter. We observed that once a message starts being disseminated, the number of new users it reaches heavily depends on the network type and the characteristics of the further messages (retweet/quote/quote retweet), which are used for dissemination. We identified certain properties, which seem to be crucial for reaching a large number of new users on Twitter, depending on the topic of discussion. A paper is currently under preparation.

7 Coupling

7.1 Status of weather and climate data sources and models

This section summarizes the latest updates in terms of coupling and dataset available with respect to the weather and climate data produced by ECMWF.

Urban Air Pollution coupling

Most of the effort regarding the coupling for the Air Pollution pilot has focused on the development of a postprocessing pipeline needed to satisfy operational and computational requirements as described in D5.7 [29]. This pipeline is based on a VM deployed in the ECMWF EWCloud infrastructure where a postprocessing tool is available to retrieve raw real-time forecast data and produce vertically interpolated data used by the UAP simulation to build high resolution urban air pollution maps.

This section aims at describing in detail the postprocessing tool, called *interpolator*, built for this workflow.

The output of ECMWF weather forecast model (Integrated Forecasting System - IFS) is produced on vertical "model levels" which are defined in terms of pressure as opposed to the altitude. To represent the vertical variation of the variables in the model, in the current operational version of the model (47R3) the atmosphere is divided into 137 layers up to 0.01 hPa (about 80 km). These layers are defined by the atmospheric pressures at the interfaces between the layers. The vertical resolution is highest in the planetary boundary layer and lowest in the stratosphere and lower mesosphere.

Outside of the meteorological and climatological domains, data is not used on these specific model levels, so the interpolation of data into altitude in meters is needed. This is the case for the Urban Air Pollution pilot application.

For the interpolation of the variables into altitude in meters, ECMWF's Python library Metview is used. During the course of HiDALGO project, the function that does this calculation was added to the Metview library and is the base of the *interpolator* tool.

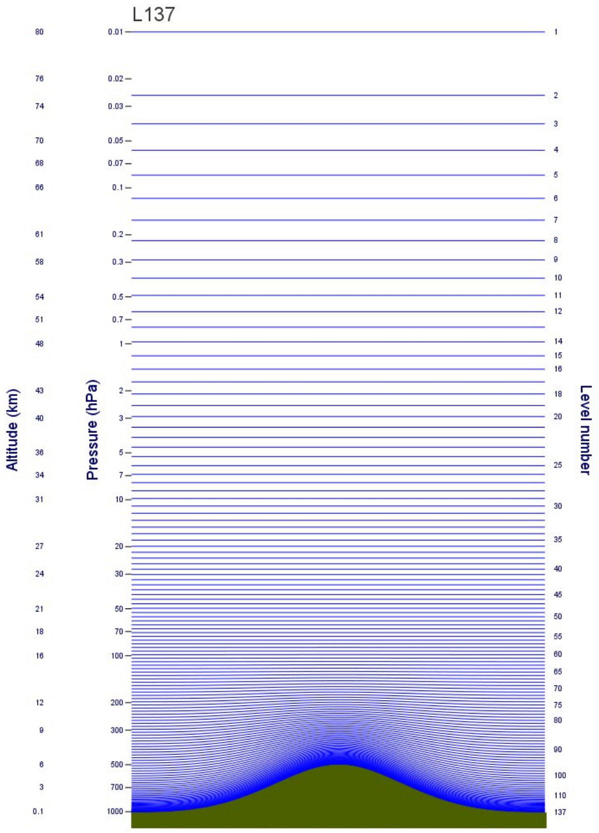


Figure 22: IFS 137 Model levels distribution

The interpolator tools is a CLI python script that allows the user to specify the following options:

- User_name and password to use to authenticate to the WCDA API. The can be the user Hidalgo Keycloak credentials.
- Date, time and geographical area for which to retrieve the forecast data
- Meteorological parameters to retrieve
- Number of steps to retrieve. Step is a measure of time, step=0 means forecast data related to current time, step=1 means forecast data related to 1h in the future and so on.
- Endpoint where notifying the completion of the execution
- Type of forecast data to retrieve, high resolution (HRES) or ensemble forecast. For the latter the number of members can be specified.

In terms of workflow, the interpolator executes the following steps:

1. Authenticate user credentials with Hidalgo Keycloak
2. Validate and transform the user inputs
3. Map the user request into WCDA requests

4. Submit the requests to the WCDA
5. Wait until data is available, download all GRIB files locally
6. Execute a vertical interpolation algorithm to transform model levels to heights
7. Compress the output to a single archive file
8. Upload it to the *ecmwf_vertical_interpolator* CKAN dataset
9. Check and delete any old files on CKAN, this helps to maintain the size of the dataset constant
10. Notify back to cloudify the URL of the file uploaded

Real time mode

Real time mode is activated if the user does not define the inputs date and time. In this case the interpolator executes the processing only when the next forecast data becomes available. This can be achieved thanks to the notification system Aviso described in D3.5 [2]. The overall workflow in this case is the following:

1. Authenticate user credentials with Hidalgo Keycloak
2. Validate and transform the user inputs
3. Map the user request into an Aviso listener configuration
4. Wait until Aviso returns a notification
5. Map the notification into WCDA requests
6. Submit the requests to the WCDA
7. Wait until data is available, download all GRIB files locally
8. Execute a vertical interpolation algorithm to transform model levels to heights
9. Compress the output to a single archive file
10. Upload it to the *ecmwf_vertical_interpolator* CKAN dataset
11. Check and delete any old files on CKAN, this helps to maintaining the size of the dataset constant
12. Notify back to Cloudify the URL of the file uploaded

Benchmarking

One of the main motivations behind the implementation of the interpolator tool in the EWCloud was to drastically improve the performance of the UAP workflow in terms of computation time and data transfers. This section aims at showing the progress achieved. The overall times reported correspond to the time taken by the interpolator workflow described above. Among all its steps, the WCDA retrieval time and the vertical interpolation algorithm are the major contributors. The effort undertaken was therefore aimed at minimising these steps. Note that the times reported are average results as single runs may be biased by the load experienced by the WCDA API and the underneath ECMWF archive system that cannot be isolated for the sake of this benchmark. The VM in the EWCloud used as part of the UAP workflow and for this benchmarks is an 8 core machine with 32 GB of RAM.

Table 5 shows the results for the simulations using only high resolution forecast data. Each row represents a specific run for a certain number of steps and a geographic area expressed in degrees of latitude x degrees of longitude. The runs reported show how the computation time increases along these 2 dimensions. They are also representative cases of user requests.

A noteworthy result is the large data size reduction between the raw data returned from WCDA and the data produced by the interpolator. This translates in a reduced data transfer time to the CKAN location used by the UAP simulation in the last phase of the workflow. The advantage in this case is however negligible given the small data size retrieved from WCDA.

HRES Run	Interpolator workflow time	WCDA data size	Output file size
0.1x0.1 - 1 step	**00:01:10**	708.9 KiB	2.3 KiB
1x1 - 1 step	**00:01:25**	837.9 KiB	16.2 KiB
10x10 - 1 step	**00:01:55**	11.4 MiB	1.1 MiB
1x1 - 2 step	**00:02:20**	1.6 MiB	32.4 KiB
1x1 - 24 step	**00:26:15**	19.6 MiB	377.6 KiB

Table 5: Benchmark for high resolution forecast over different area size in degrees and number of steps

Ensemble Run	Interpolator workflow time	WCDA data size	Output file size
0.2x0.2 - 1 step – FP	**00:54:00**	38.5 MiB	65.2 KiB
1x1 - 1 step – FP	**00:56:30**	37.6 MiB	260.7 KiB
10x10 - 1 step – FP	**01:23:20**	176.6 MiB	14.4 MiB
1x1 - 2 step – FP	**00:56:50**	115.6 MiB	795.4 KiB
1x1 - 24 step – FP	**02:37:00**	924.5 MiB	7.2 MiB
1x1 - 24 step – SP	**08:44:30**	924.5 MiB	7.2 MiB
1x1 - 24 step – S	**16:15:00**	924.5 MiB	7.2 MiB

Table 6: Benchmark for ensemble forecast over different area size in degrees, number of steps and type of workflow (FP=Fully Parallel, SP=Semi Parallel, S=Sequential)

Table 6 shows the results of ensemble forecast processing. Similarly to the high resolution case the runs reported are in function of the geographical area and the number steps. The benchmark is reported also in function of the type of workflow used to process the ensemble data. 3 type of workflows have been implemented to parallelise the computation, they represent incremental improvements in terms of computation time:

- Sequential (S), all data is retrieved from the WCDA using parallel requests and once completed each step is processed one at a time. This implementation reflects initial concerns over Metview library not being multi-process safe.
- Semi Parallel (SP), all data is retrieved from the WCDA using parallel requests and once completed all steps are processed in parallel.
- Fully Parallel (FP), for each step an independent process retrieves the data for this step from the WCDA and once completed it processes the step. All the processes run in parallel. This is the optimal way to run the ensemble processing as shows by the results.

The processing of the ensemble forecasts deals with a much larger amount of data. The ECMWF Ensemble Prediction System represents uncertainty in the initial atmospheric conditions by creating a set of 50 forecasts (ensemble members) starting from slightly different states that are close, but not identical, to our best estimate of the initial state of the atmosphere (the control). Each forecast is based on a model which is close, but not identical, to our best estimate of the model equations, thus representing also the influence of model uncertainties on forecast error.

The optimisations performed by the interpolar script in terms of processing and data transfer are in this case remarkable and essential for the feasibility of the overall UAP workflow, specifically in the case of a real time scenarios where the simulation has to terminate before the final data lose their values because too old for activating any actions from the local authorities.

Migration coupling

Global simulations of river discharge produced by Global Flood Awareness System (GloFAS) at ECMWF is used by Human Migration Pilot application. It has been available through Copernicus Climate Data Store (https://cds.climate.copernicus.eu/cdsapp#!/dataset/cems-glofas-forecast?tab=overview) using web download tool and CDSAPI for download.

This dataset was available in netCDF4 format until December 2020. In order to be able to archive this forecast and reanalysis data in ECMWF's MARS (Meteorological Archival and

Retrieval System) archive, the format was changed to meteorological GRIB2 (https://codes.wmo.int/grib2) binary data format.

The change in the format resulted in a small change in the RestAPI for retrieving data, therefore the scripts to download and process river discharge data for the Human Migration pilot were updated.

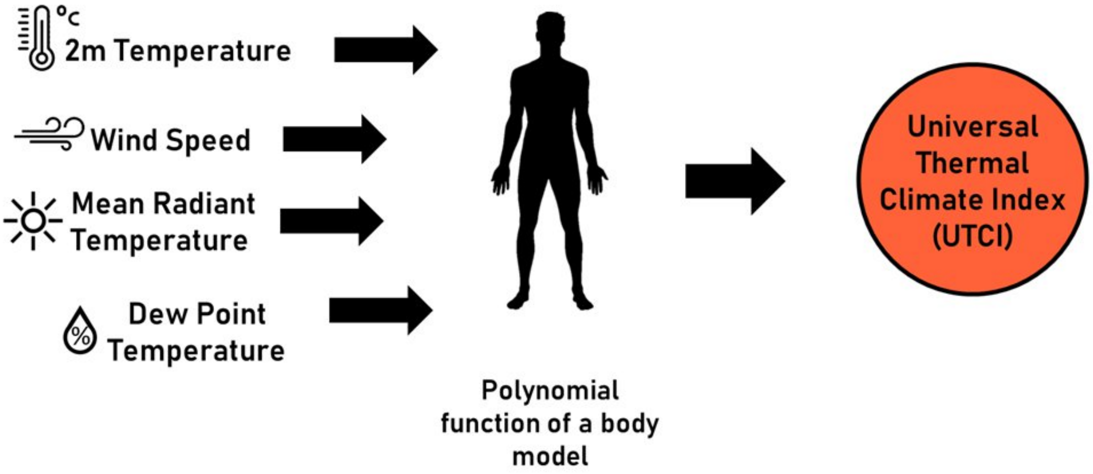
Universal Thermal Climate Index

Extreme high temperatures can have serious impacts on human health. They are most common during the heat waves which have been increasing in intensity, duration and frequency in the last decades, with a growing evidence of them being more intense, long lasting and more frequent in the following decades, making them a unique and complex global challenge. However, it cannot be considered isolated from other global challenges, which is why ECMWF has decided to put an effort in the development of the Universal Thermal Climate Index under HiDALGO project, with the aim of offering this data for the use of the Pilot Applications as well as global challenge applications beyond HiDALGO.

The Universal Thermal Climate Index (UTCI) models the response of the average human body to the outside thermal environment in terms of 2 m air temperature, mean radiant temperature, relative humidity, and 10 m wind speed. It is an important thermal comfort index because it has been shown to be able to model thermal extreme related mortality in Europe [30]. In addition, it captures characteristics of heat stress at an international scale [31] and can be used to forecast heat extremes [32].

It was originally developed as part of an international biometeorological commission and then European COST (Cooperation in Science and Technical Development) Action 730, brought to completion in 2009. Then, as part of the Anywhere project the UTCI was developed for use by ECMWF and used to output ERA-5 HEAT, this was the first gridded thermal comfort index dataset available international.

As part of the HiDALGO project, the UTCI development was assessed and improved. Moreover, the python package *thermofeel* was released as open source (https://github.com/ecmwf-projects/thermofeel/releases/tag/1.0.0). This package includes the UTCI calculation along with other notable thermal indices such as the Wet Bulb Globe Temperature.


°C
2m Temperature
Wind Speed
Mean Radiant Temperature
Dew Point Temperature
%
Polynomial function of a body model
Universal Thermal Climate Index (UTCI)

7.2 Status of telecommunication data sources

Telecommunication data provided by MOON are intended to be used by the migration and social network pilots. D4.3 [1] provides a detailed description of the data format and data layout.

Status report of the coupling mechanisms

REST API is developed to allow dynamic coupling of real-time telecommunication data. It delivers data sets from the MOON data servers to the migration and social network pilots running on the HiDALGO HPC centre. The delivered CDRs are aggregated according to the migration (plain) or social network (social) pilots. The provided data is handed over to other partners within HiDALGO in anonymized form, and no personal information can be extracted from them, except for Moonstar who stores the original data. The anonymization techniques are described in detail in D8.1 [33]. The API delivers the requested data in plain csv or zip-compressed csv. Deliverable 3.5 [2] provides a complete description of the API.

Migration coupling

The purpose of telecommunication data is to model the journey of refugees from conflict zone and the different camps. It will be used to validate the simulation results of the migration pilot. D4.3 [1] reported that datasets containing mobile calls do not provide spatial information about the journey of refugees. They only provide limited information like the call duration (total number of minutes) per day or the number of calls each user has been made. For this reason, a large data set related to the Syrian conflict is accessible from the MOON servers. It consists of home calls between the main cities in Syria, Turkey and Iraq during the time period from January 1, 2016 until June 1, 2020. The GPS location of the cities of the caller (A_NUMBER) and callee (B_NUMBER) are added to the data.

Using this way, data will be aggregated to the regional level to match the census migration data collected from conflict and camp locations. Time series of the number of calls per day and the total number of minutes per day will be plotted with the flow of refugees in the same region. This may allow the validation of the simulation results concerning the Syrian conflict.

Social network coupling

Telecommunication data are dynamically provided to the social network pilot in a special format. The CDRs are aggregated by B_numbers, see Table 7.

B_numbers	Description
(B_number_1, B_network)	[(A_number_1, country, A_network, number of calls), (A_number_2, country, A_network, number of calls), ….]
(B_number_2, B_network)	[(A_number_1, country, A_network, number of calls), (A_number_2, country, A_network, number of calls), ….]
…	…

Table 7: An example of aggregated CDRs for the social network

Coupling the telecommunication data with the social network pilot will be investigated in the forthcoming weeks of the project, between the submission of this deliverable and the final review.

Scalability assessment of the couplings

Access times to telecommunication data and scalability assessment will be possible only when the coupling mechanisms will be in place.

7.3 Status of traffic data and local air quality sensor data

For the traffic data, the current method which is still under development includes following steps:

1. Number and distribution of trips are determined from historical trip data and predicted traffic monitoring measurements
2. Traffic light scheduling scenarios are set in SUMO simulation and KPIs like Mean Travel Time are calculated from ran SUMO simulations (Figure 23)
3. Best option is chosen and AQ verified by full UAP

4. Physical traffic lights are set remotely by the script to Traffic Light Monitoring and Control System with Magyar Közút (HU Public Road Agency); *ongoing*

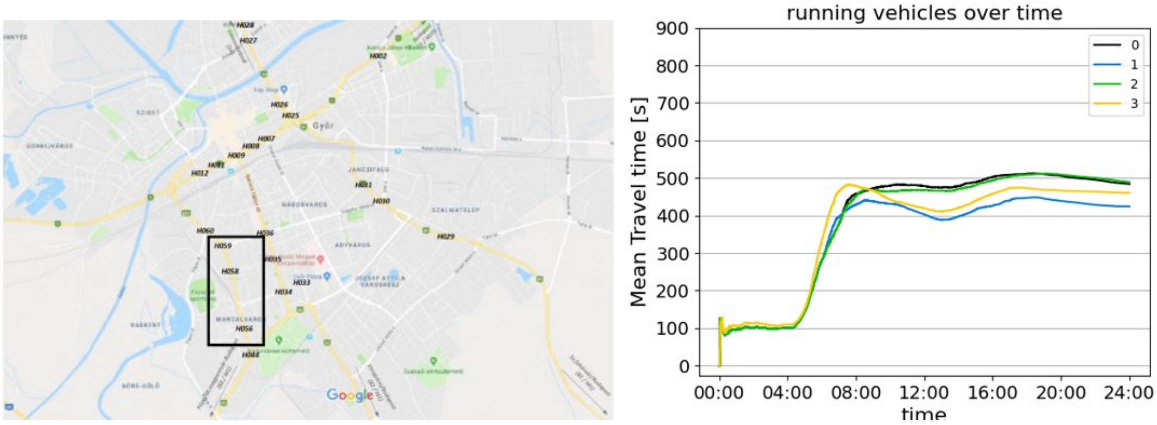


Figure 23: Traffic Light Monitoring and Control System and Mean Travel Time

Ongoing works in this part are: Plugin to the existing traffic light monitoring system (JTR) – easy implementation to all cities with similar traffic light monitoring (e.g. each Hungarian cities) and Completion of the traffic control implementation in real traffic. Any topics of the UAP has many challenges and business opportunities, open for collaboration.

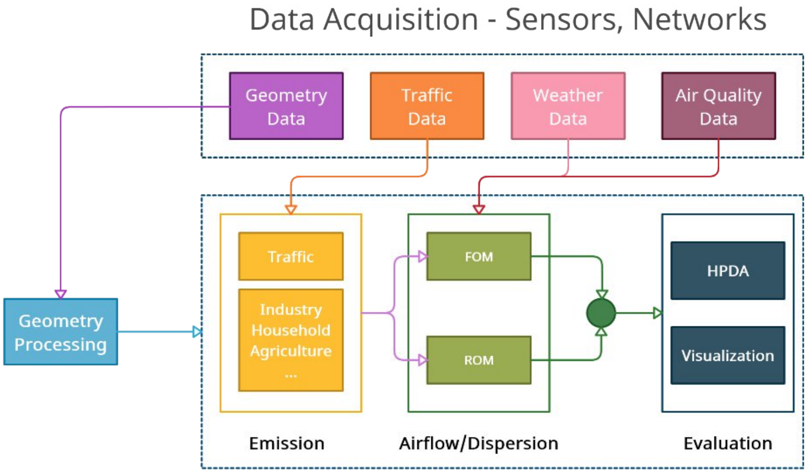


Figure 24: The components of the UAP workflow

Status report of the coupling mechanisms

The components of the coupling in UAP are depicted in Figure 24. Data acquisition component includes: Geometry data processing, traffic data, weather data, and air quality data. For instance for Geometry data which can be automatically processed (with own code) the Inputs are bbox coordinates of any cities on Open Street Map and resolution. The output contains

3D geometry and Computational mesh. The other components are Traffic data and simulation, emission computations.

Case 1: for the HiDALGO demo area, city of Győr

- Loop & camera + radar sensor network in the HiDALGO demo area with camera plate recognition and vehicle classification
- Traffic simulation with SUMO with data assimilation, done by Bosch

Case 2: other cities

- Traffic (or emission) data from stakeholders (top-down, simulation results by the city, etc.)
- Optionally: configured random simulation with SUMO

Emission: using Copert 5 standard tool for the traffic and fleet data

Weather data

One if the problems we are dealing with is the Coupling UAP and a regional weather model, e.g. WRF into ECMWF > WRF > UAP. It includes Scalable WRF models for ensemble simulation and Challenges in UQ (uncertainty quantification) and data assimilation.

- Forecast simulations by ECMWF, downloaded through Polytope API and EWCloud: Performance is tested: nearly the same access time
- Resolution: cca. 10km x 10km
- Retrieved data
  - Wind and pressure sampled at vertical lines (CFD boundary)
  - Planetary Boundary Layer Height
  - Surface values of 10 variables (for the AI prediction of sensor data)
  - Ongoing: ensemble data for uncertainty quantification

AQ data

- HiDALGO Immission Measurement Boxes (IMB units) provided by Bosch
  - 2008/50/EC certified measurement of NO2, O3, PM2.5, PM10 (AQ), and p, T, rH (weather)
  - 5 IMB units since 2020-11-20
  - Data are available through Bosch Cloud by API
- National/EU AQ monitoring stations
  - Weather and AQ Data, retrieved from the web
- Copernicus Climate Data Store (CDS)
  - Background and surface AQ data

Scalability assessment of the couplings

Optimization of numerical solver configurations

- PPGR solver for pressure equation (computation and MPI-communication done in parallel)
- Internal iterations according to limited accuracy (new feature of OpeFOAM)
- Limiting corrections due to good mesh orthogonality (octree mesh)

Improvement of domain decomposition

- Multilevel domain decomposition adapted to the computer architecture
- Cell index reordering at lowest level to limit leaps in memory access

Limiting of I/O

- Turning off default features that frequently access disk
- Coalesced domain decomposition to limit number of files

Continuous reading of emission data limits memory consumption

- From the large amount of per cell emission data the only necessary time steps are read and stored

Benchmarking results are presented in Figure 25 and Table 8.

- Computer: Hawk, HLRS (5632 x 2 x 64 core AMD Rome 7702), 1 node=128 core
- Unsteady simulation of 1 hour; mesh sizes 728k, 3.4M, 14M

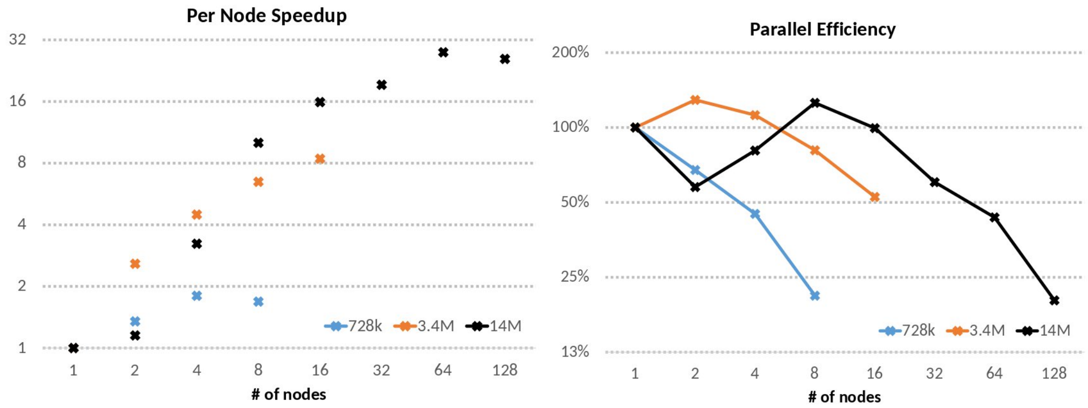


Figure 25: Node Speedup and Parallel Efficiency for three different mesh sizes

Mesh size | resolution	Time to simulate 1 physical day	Time to simulate 1 physical year	Estimated Flops
728k (Gyor) | 4 meter	0.3 hour (on 4 nodes)	4.5 days (on 4 nodes)	16 TFLOPS

Mesh size | resolution	Time to simulate 1 physical day	Time to simulate 1 physical year	Estimated Flops
3.4M (Gyor) | 2 meter	1.5 hour (on 16 nodes)	23.2 days (on 16 nodes)	64 TFLOPS
14M (Gyor) | 1 meter	12.5 hour (on 64 nodes)	189,8 days (on 64 nodes)	256 TFLOPS

Table 8: Benchmarking results

7.4 Status of Conflict Modelling of Migration pilot

Within the migration pilot, the forecasting of population displacements driven by conflicts comes with an essential prerequisite: the forecasting of the conflict itself. This is a tricky aspect of the modelling, because a completely perfect conflict forecasting tool is actually ethically undesirable, as it is highly prone to dangerous misuse. Instead we aim for a modelling approach that generates reasonably realistic conflict progressions in a stochastic way, and using fairly trivial assumptions.

Previously, we have presented a particularly primitive mechanistic approach (named Flare), which allows users to generate randomized conflict progressions. This work is presented by Groen et. al. [34], and shows how conflict models can be used to create simulations with conflict-driven uncertainties. Within the context of HiDALGO we revised the Flare model, moving away from a non-validated mechanistic approach to a validated and more data-driven approach.

This new version of Flare implements a rule set majority voting system, where multiple rule sets “vote” on the same location to determine whether or not there is a conflict event. This mechanism allows the code to discover and apply complex composite algorithms. The ruleset is primarily driven by population numbers of given locations, and considers a location’s share of the national / regional population rather than the actual number of people. Our choice for using relative population values is motivated by the fact that such a ruleset can be more widely re-applied (e.g. to countries with smaller or larger populations.

We gathered, processed and trained rule sets on six complete conflicts over the course of the 1990s. The resulting rule set was then applied to a seventh conflict, which took place in Burundi.

In Figure 26, we report on the validation performance of our new Flare code for the Burundi conflict, using three rulesets. The first ruleset uses fractions of national population, the second ruleset uses fractions of regional population, while the third ruleset combines both of the previous rulesets. In general we find that Flare underpredicts the number of conflicts in this case, but that ~50% of the conflicts predicted also occurred in real life.

Going forward, our priority is to automate the training of the model (which is still largely manual), and to validate Flare against a wider range of conflict situations.

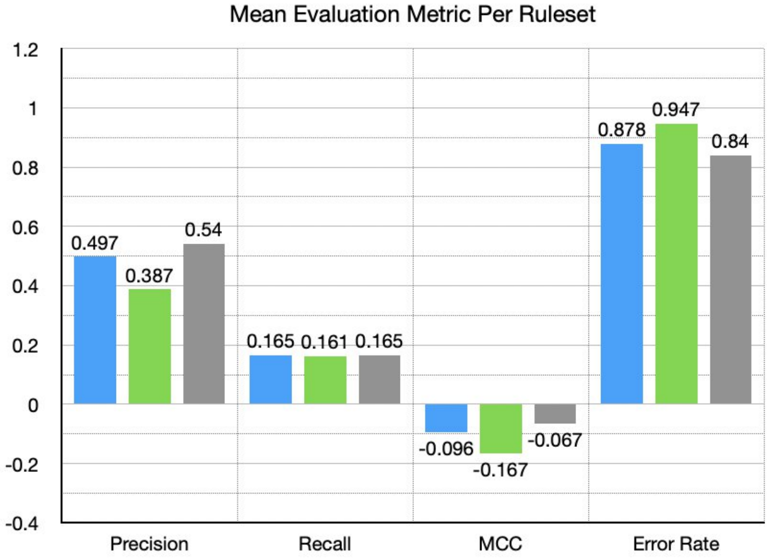


Figure 26: Validation performance of Flare when trained on six conflicts and applied to a conflict in Burundi. Here blue bars are for national population fractions, green bars are for regional population fractions, and grey bars are for a hybrid approach that task.

Status report of the coupling mechanisms

The coupling between the conflict model (Flare) and Flee is done using a file called conflicts.csv. The coupling mechanism has been in place since the first year of HiDALGO, and did not require any updates since its initial version documented here [34].

Scalability assessment of the coupling mechanism

The coupling between the conflict model (Flare) and Flee is light weight, in that the conflicts.csv file exchanged between the code is typically less than 1MB in size.

7.5 Status of Hospital Model integration with the Covid-19 Pilot

One additional type of coupling we performed within the context of HiDALGO is the integration of FACS with a local hospital model. Within the context of the STAMINA project (https://www.stamina-project.eu) we established a separate simulation code for modelling hospital bed occupancy and logistics. This code, named CHARM (Anagnostou et al., *in prep.*), is still under the final stages of testing and we expect a release under the STAMINA project within the coming year.

Status report of the coupling mechanisms

As part of our collaboration with NHS Trusts in West London, we are required from time to time to model the events not only in the hospital catchment area (using FACS), but also within the hospital itself (using CHARM). To enable an integrated forecast using both solvers, we have modified the output of FACS such that individual hospitalisations are logged by the simulator. These hospitalisations are stored in a CSV file format, which is then loaded in by CHARM to model the arrival of coronavirus patients within the hospital setting.

Columns	Format	Description
t	Integer	Arrival time of the hospitalised COVID-19 patients.
x	Signed float	Home location of the infected agent/person (x coordinate).
y	Signed float	Home location of the infected agent/person (y coordinate).
age	Integer	Age in years.

Table 9: Format of the out_hospitalisations.csv output file in FACS, specifically developed to enable coupling with hospital model. The file contains one row per hospital admission

The modifications to FACS made in this project are made publicly available in the master branch of the GitHub repository, allowing other hospital modelling codes also to load hospital admissions output from FACS.

Scalability assessment of the coupling mechanism

The coupling of FACS to CHARM relatively light weight in terms of computational and I/O burden. Typically, we extract hospital admissions from a single representative run for use in CHARM. The size of such a data set is equal to approximately 40 bytes per hospital admissions, which results in a coupling file of less than 1MB in size per simulation run.

8 Conclusions

In this deliverable we have presented the current ("final") status of the pilot applications in HiDALGO, as well as the coupled data sources and models in the project. Our pilots include mature and scalable applications in human migration, urban air pollution and social network analysis, as well as an emerging fourth pilot on COVID-19 spread modelling.

In migration, we have leveraged the scalability of the Flee solver (already achieved earlier in the project) to widen the applicability and uptake of the code. For example, we established a conflict forecasting tool (named FLARE) that can extrapolate conflicts based on training data, enhanced Flee to accurate account for precipitation and flooding within its simulations. We also applied Flee directly to produce forecasting reports for Tigray, Ethiopia for our collaborators at the Save The Children NGO, which more recently led to the Institute of Migration joining our collaborative effort on this.

For urban air pollution, we dedicated significant effort to make our simulations both more robust and less expensive to run. Through our new model order reduction approaches we can save up to two orders of magnitude on run-time, allowing us to do rigorous ensemble analysis at a tractable cost. Here, too, we have widened the applicability and uptake of the code, showcasing its use in diverse cities such as Gyor, Graz, Stuttgart and most recently Antwerp. In addition, we have established a tight development collaboration with Bosch and initiated a new application direction where our simulations can directly guide the operation of real-world traffic lights.

For social networks, we have been able to establish a mature social networks simulation which integrates Twitter as a direct data source. We have shown that our approach can be used to analyse major social network trends on demand, and that our simulator reproduces key phenomena that we also observe in the Twitter data. In terms of performance, we have shown that our eigenvalue histogram calculator scales linearly to 10,000s of cores, while our newly developed SN-simulator scales near-linearly to about 500 cores per run.

For COVID-19, we have been able to bring the Flu And Coronavirus Simulator (FACS) from first code line to uptake by external hospital stakeholders in less than a year. The code was developed rapidly in response to the epidemic emergency, and during Y3 of the project we focused on bolstering the code base, and widening its uptake. FACS has now been taken up by a range of external parties, and has been retroactively included (the grant agreement was amended to include it) in the pandemic trials within the EU-funded STAMINA project because of its newfound (HiDALGO-powered) existence. The M37 timing of this deliverable is slightly unfortunate in that we did not yet finalize the testing of a parallelized FACS version, but our aim is to have this completed by the end of the project (M39).

In terms of coupling we have covered a wide range of models and data sources that are now integrated with the four pilots. Particular highlights in this recent period are the much more extensive and sophisticated integration of ECMWF data within both the migration and urban air pollution pilots, the integration of Twitter data with the social networks pilots, and the integration of traffic sensors (and even a traffic light) with the urban air pollution pilot. As expected, some coupling mechanisms have shown to be more beneficial than others. For instance, the integration of telecommunications data in migration has shown to be less fruitful than anticipated: this is primarily due to the logistical and ethical difficulties around obtaining such data systematically across a location-specific, country-wide scope, and doing so over a prolonged time period. At the same time the migration model integration with ECMWF data was highly successful in the South Sudan context, and we now apply it also to better assess location connectivity in the Mali conflict. As another positive example, the integration of UAP with traffic sensor networks has found clear applicability in cities outside of Gyor. Although no coupling was technically required in the GA for the COVID-19 pilot, we did incorporate a coupling with a hospital bed model in response to needs from local National Health Service Trusts in London, UK.

In terms of validation, we have presented a wide range of results showing that our simulation results largely correspond with the observations. Importantly, we have managed to extend the number of validation settings in particularly for the migration and urban air pollution pilots, showcasing the reusable nature of our approaches. As for future work, many of the codes presented in the deliverable have been taken up by external research projects, and we expect that further development can be maintained beyond HiDALGO through these projects.

References

[1] HiDALGO, “D4.3 - Implementation Report of the Pilot Applications Year 2,” Robert Elsässer, 2020.

[2] HiDALGO, “D3.5 - Final Report on Benchmarking, Implementation, Optimisation Strategies and Coupling Technologies,” Marcin Lawenda, 2021.

[3] K. Deb, S. Agrawal, A. Pratap and T. Meyarivan, “A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II,” in *International conference on parallel problem solving from nature*, 2000.

[4] K. Deb and H. Jain, “An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints,” *IEEE transactions on evolutionary computation*, vol. 18, no. 4, pp. 577--601, 2013.

[5] Q. Zhang and H. Li, “MOEA/D: A multiobjective evolutionary algorithm based on decomposition,” *IEEE Transactions on evolutionary computation*, vol. 11, no. 6, pp. 712-731, 2007.

[6] Q. D. Team, *QGIS Geographic Information System*, Open Source Geospatial Foundation, 2009.

[7] L. E. Veen and A. G. Hoekstra, “Easing multiscale model design and coupling with MUSCLE 3,” in *International Conference on Computational Science*, 2020.

[8] “ International Organization for Migration,” [Online]. Available: https://www.iom.int/.

[9] “United Nations Office for the Coordination of Humanitarian Affairs,” [Online]. Available: https://www.unocha.org.

[10] “IT Tools and Methods for Managing Migration Flows,” [Online]. Available: https://www.itflows.eu/.

[11] “Smart support platform for pandemic prediction and management,” [Online]. Available: https://stamina-project.eu.

[12] D. Suleimenova, H. Arabnejad, W. Edeling and D. Groen, “Sensitivity-driven simulation development: a case study in forced migration,” *Philosophical Transactions of the Royal Society A*, vol. 379, no. 2197, p. 20200077, 2021.

[13] “Poznan Supercomputing and Networking Center,” [Online]. Available:

https://www.psnc.pl/.

[14] C. Schweimer, B. C. Geiger, M. Wang, S. Gogolenko, I. Mahmood, A. Jahani, D. Suleimenova and D. Groen, “A route pruning algorithm for an automated geographic location graph construction. Scientific reports,” *Scientific reports*, vol. 11, no. 1, pp. 1-11, 2021.

[15] A. Jahani, H. Arabnejad, D. Suleimanova, M. Vuckovic, I. Mahmood and D. Groen, “Towards a Coupled Migration and Weather Simulation: South Sudan Conflict,” in *International Conference on Computational Science*, 2021.

[16] I. Mahmood, H. Arabnejad, D. Suleimenova, I. Sassoon, A. Marshan, A. Serrano-Rico, P. Louvieris, A. Anagnostou, S. J. Taylor, D. Bell and others, “FACS: a geospatial agent-based simulator for analysing COVID-19 spread and public health measures on local regions,” *Journal of Simulation*, pp. 1-19, 2020.

[17] HiDALGO, “D6.5 - Intermediate Report on Requirements, Components and Workflow Integration,” Sergiy Gogolenko, 2020.

[18] L. Kornyei, Z. Horvath, A. Ruopp, A. Kovacs and B. Liszkai, “Multi-scale Modelling of Urban Air Pollution with Coupled Weather Forecast and Traffic Simulation on HPC Architecture,” in *The International Conference on High Performance Computing in Asia-Pacific Region Companion*, 2021.

[19] E. Bakshy, J. M. Hofman, W. A. Mason and D. J. Watts, “Everyone's an influencer: quantifying influence on twitter,” in *Proceedings of the fourth ACM international conference on Web search and data mining*, 2011.

[20] C. Kuehn, E. A. Martens and D. M. Romero, “Critical transitions in social network activity,” *Journal of Complex Networks*, vol. 2, no. 2, pp. 141-152, 2014.

[21] M. Cinelli, G. D. F. Morales, A. Galeazzi, W. Quattrociocchi and M. Starnini, “The echo chamber effect on social,” *Proceedings of the National Academy of Sciences*, vol. 118, no. 9, 2021.

[22] “Numba,” [Online]. Available: https://numba.pydata.org/.

[23] M. C. Cario and B. L. Nelson, “Modeling and generating random vectors with arbitrary marginal distributions and correlation matrix,” Citeseer, 1997.

[24] S. Bansal, S. Khandelwal and L. A. Meyers, “Exploring biological network structure with clustered random networks,” *BMC bioinformatics*, pp. 1-15, 2009.

[25] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart

and D. Terry, “Epidemic algorithms for replicated database maintenance,” in *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, 1987.

[26] H. W. Hethcote, “The mathematics of infectious diseases,” *SIAM review*, vol. 42, no. 4, pp. 599-653, 2000.

[27] G. a. B. P. Bankhamer, F. Biermeier, R. Elsässer, H. Hosseinpour, D. Kaaser and P. Kling, “Fast Consensus via the Unconstrained Undecided State Dynamics,” *arXiv preprint arXiv:2103.10366*, 2011.

[28] G. Bankhamer, R. Elsässer, D. Kaaser and M. Krnc, “Positive Aging Admits Fast Asynchronous Plurality Consensus,” in *Proceedings of the 39th Symposium on Principles of Distributed Computing*, 2020.

[29] HiDALGO, “D5.7 - Final Portal Release and System Operation Report,” Li Zhong, 2021.

[30] C. Di Napoli, F. Pappenberger and H. L. Cloke, “Assessing heat-related health risk in Europe via the Universal Thermal Climate Index (UTCI),” *International journal of biometeorology*, vol. 62, no. 7, pp. 1155--1165, 2018.

[31] C. Brimicombe, C. Di Napoli, R. Cornforth, F. Pappenberger, C. Petty and H. L. Cloke, “Borderless heat hazards with bordered impacts,” *Earth's Future*, vol. 9, no. 9, p. e2021EF002064, 2021.

[32] F. Pappenberger, G. Jendritzky, H. Staiger, E. Dutra, F. Di Giuseppe, D. Richardson and H. Cloke, “Global forecasting of thermal health hazards: the skill of probabilistic predictions of the Universal Thermal Climate Index (UTCI),” *International journal of biometeorology*, vol. 59, no. 3, pp. 311-323, 2015.

[33] HiDALGO, “D8.1 - POPD Requirements No. 1,” Ivan Zaldivar, 2019.

[34] D. Groen, D. Bell, H. Arabnejad, D. Suleimenova, S. J. Taylor and A. Anagnostou, “Towards modelling the effect of evolving violence on forced migration,” in *2019 Winter Simulation Conference (WSC)*, 2019.