Applied Intelligence (2023) 53:4063–4098
https://doi.org/10.1007/s10489-022-03605-1

Explaining deep reinforcement learning decisions in complex multiagent settings: towards enabling automation in air traffic flow management

Theocharis Kravaris[1] · Konstantinos Lentzos[1] · Georgios Santipantakis[1] · George A. Vouros[1] · Gennady Andrienko[2,3] · Natalia Andrienko[2,3] · Ian Crook[4] · Jose Manuel Cordero Garcia[5] · Enrique Iglesias Martinez[5]

Accepted: 3 April 2022 / Published online: 6 June 2022

Abstract

With the objective to enhance human performance and maximize engagement during the performance of tasks, we aim to advance automation for decision making in complex and large-scale multi-agent settings. Towards these goals, this paper presents a deep multi agent reinforcement learning method for resolving demand - capacity imbalances in real-world Air Traffic Management settings with thousands of agents. Agents comprising the system are able to jointly decide on the measures to be applied to resolve imbalances, while they provide explanations on their decisions: This information is rendered and explored via appropriate visual analytics tools. The paper presents how major challenges of scalability and complexity are addressed, and provides results from evaluation tests that show the abilities of models to provide high-quality solutions and high-fidelity explanations.

Keywords Air traffic management · Multi-agent deep reinforcement learning · Interpretability · Stochastic decision trees · Explainability · Visualization

1 Introduction

Complex and large-scale multi-agent problem settings typically involve a large number of self-interested agents with interacting, and potentially conflicting decisions/actions, and dynamic avalanches of actions' effects in space and time, affecting others. Such settings appear in various real-life domains (urban traffic congestion, air traffic management and network routing), with emerging challenges. This class of problems includes interesting real-world congestion problems, which have drawn much attention in the AI and autonomous agents research (e.g. [1–5]) for at least two decades [6] and have been the focus of game theoretic models for much longer [7, 8].

Aiming to contribute to the automation of operations in a real-life complex and large-scale multi-agent setting where congestion problems arise, we need to meet not only domain-specific objectives, but also objectives regarding human performance and engagement. In this context, a new important challenge emerges, regarding trustworthiness in the system, in the sense that operators should be comfortable relinquishing control to it, given appropriate explanations on system's decision making [9]. Major factors affecting trustworthiness include quality of automated solutions, as well as quality of explanations provided. As far as we know, while many works address the computation of problems' solutions in complex multi-agent settings, there is not any work that addresses the dual challenge of providing (a) qualitative solutions in large scale settings and (b) coherent and concise explanations regarding agents' joint decision making in large scale and complex settings.

In this work we address the challenging issues of scalability and complexity towards advancing automation in real-world multi-agent settings with thousands of agents, aiming to (a) compute qualitative solutions to congestion

✉ George A. Vouros
georgev@unipi.gr

[1] University of Piraeus, Piraeus, Greece

[2] Frauhofer Institute IAIS, Sankt Augustin, Germany

[3] City University of London, London, UK

[4] ISA Software, Paris, France

[5] CRIDA, Madrid, Spain

problems and (b) provide explanations on how individual agents' decisions jointly affect their common setting.

Specifically, in the air-traffic management (ATM) domain, *demand and capacity balancing* problems (DCB) are a kind of congestion problems that arise naturally whenever demand of airspace use exceeds capacity, resulting to *"hotspots"*. Hotspots are resolved via capacity management or flow management solutions, including regulations that generate delays and re-routings to flights, causing unforeseen effects for the entire system, and increasing uncertainty regarding the scheduling of (ground and airspace) operations. For instance, flight delays cause the introduction/increase of time buffers in operations' schedules, and may accumulate demand for resources within specific periods. These are translated into costs and negative effects on airlines' reliability, customers' satisfaction and environmental footprint.

Increased demand for airspace use is the major factor for unprecedented measures applied to flights (over 90% in some airspaces) [10]. It got significantly worse in 2018 [11] when delays across Europe more than doubled, due to the increase in traffic, among other factors. In general, all performance analysis and studies lead to the idea that the ATM system in the post-Covid-19 era was very close to, or already at, a saturation level. These issues, in conjunction to the foreseen increase in air traffic [12, 13] impose the need for the assessment and minimization of measures applied to flights at the *"pre-tactical"* phase of operations (i.e. from several days to few hours before operations), so as to satisfy as much as possible airspace users' plans and bring no surprises to their operations. This contributes at increasing the predictability of the overall ATM system, alleviating many of the negative effects.

Today, measures are imposed to flights without considering the propagated effects to the entire ATM system (e.g. to other flights and airspaces). Indeed, even at the pre-tactical phase of operations, measures are decided in a rather greedy way: Congested airspaces are regulated, and measures are applied to specific flights entering these airspaces in a first-come-first-regulated basis. This functionality is implemented by the Computer Assisted Slot Allocation (CASA) system, as also described in [14]. While this may resolve local congestion, it considers neither (a) the optimal mixture of measures one needs to apply to all flights w.r.t. their temporal and spatial interactions, nor the (b) side-effects to the inherently complex and dynamic ATM system (e.g. the emergence of hotspots and their effects on other airspace users). In practice, many hotpots remain unresolved at the pre-tactical phase, with the aim to resolve these at the tactical phase, introducing additional factors of uncertainty and inefficiency of ATM stakeholders' operations. For instance, Table 1 shows regulations imposed to flights towards resolving hotspots during 3 days in 2019: While the average delay per flight is small and the number of flights affected are not many, the hotspots in the Spanish airspace after imposing these regulations, for each of these days, are really many: In the first case almost as many as in the original situation (i.e. prior to the regulations), or even worse in the other days.

Table 1 Delay regulations imposed to flights in three days

Day	Average Delay(min)	Delayed Flights	Unresolved Hotspots
20190705	3.29	1204	99
20190708	2.93	771	81
20190714	2.70	1010	74

Indeed, solutions to hostpots may be impossible in some cases, as in the case shown in Fig. 1, where there are many hotspots for many consecutive periods during the day: Blue bars show the demand in an airspace compartment (sector LECPDEO) for different periods during a day, and the orange line indicates the 110% of the sector capacity (we explain operational terms in detail in the next section). Measures to resolve a hotspot in a specific period may cause cascade effects to the same or other sectors in different times: This imposes the need for a global view of the airspace, requiring higher levels of automation to support humans' decision making, due to the scalability and the inherent complexity of the problem.

Towards these aims, herein we continue our work [15–19] on agent-based approaches to resolving DCB problems in the ATM domain: Agents, representing flights, aim to decide on own measures, jointly with others, with respect to own preferences and operational constraints on the use of airspace, while possessing no information about the preferences and payoffs of others. However, we go beyond our previous efforts in several critical aspects: First, instead of producing a single type of solution, agents are able to produce different types of solutions with different measures for resolving DCB problems. Second, addressing the limitations of tabular Q-learning methods, we propose the use of a deep multi-agent reinforcement learning method for resolving demand - capacity imbalances in real-world ATM settings, being able to consider numerous and potentially continuous features of states, while addressing robustness, generalization and scalability issues in training the agents. Finally, we emphasize on the explainability of agents' decision making, which is supported by appropriate visual analytics tools to render and explore agents' decision making (i.e. explanation content).

Specifically, the contributions made in this paper are as follows:

- We devise a scalable multi-agent deep reinforcement learning method based on DQN, providing a policy

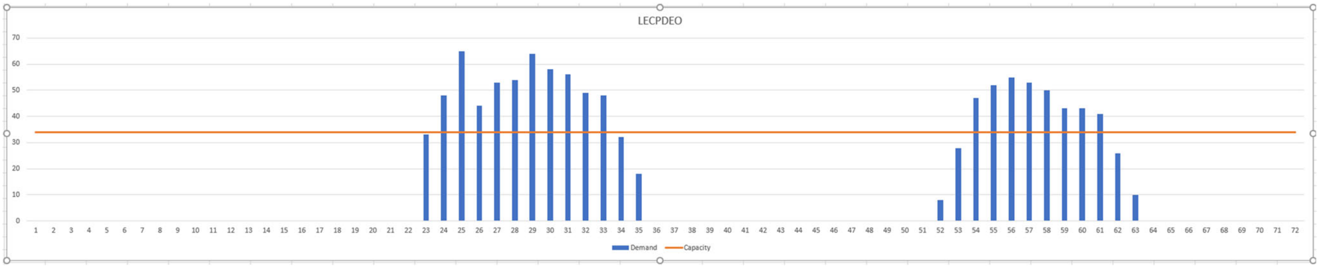


Fig. 1 Evolution of daily demand in sector LECPDEO in a singe day

model for agents to jointly decide on DCB resolution measures in an inherently complex and large-scale real-world multi-agent setting.

- We propose the use of Stochastic Gradient Trees to build interpretable models that mimic the decisions of the well-trained DQN model, providing explanation content for agents' local decisions.
- We propose visual analytics methods for rendering and exploring solutions and explanations, addressing the scale of the multi-agent task, and the complexity of the problem.
- We report on important findings and lessons learnt after validating the system with experts in real-world DCB scenarios.

Major findings of this article are that multi-agent Deep Reinforcement Learning methods (a) can scale up to the number of agents that operate in an airspace, (b) providing qualitative solutions to DCB problems, thus, imposing measures to individual flights that resolve most of the existing hotspots, while they (c) can be made explainable, towards keeping the human in the loop, enhancing situation awareness, using (d) advanced visualization methods. However, on the negative side, (e) explanation content can be made very detailed and large in volume, due to the scale and complexity of the problem, while (f) it is very difficult for a human to master the complexity of the system and fully understand the explanations provided. This is due to the inherently large number and unpredictable interactions between agents.

The structure of this paper is as follows: Section 2 presents related efforts on reinforcement learning techniques in resolving demand and capacity balancing problems, as well as related work on scalable multi-agent deep reinforcement learning and explainable deep reinforcement learning. Section 3 specifies the problem and introduces terminology from the ATM domain. Section 4 presents the problem formulation within a multi-agent MDP framework and shows that the problem can be considered as a Markov Game. Section 5 identifies critical aspects regarding explanations. Section 6 presents the deep reinforcement learning method proposed for solving the problem, while Section 7 presents the explainability paradigm advocated and the interpretation method used. Section 8 presents visual analytics tools for exploring solutions and explanations, while Section 9 presents evaluation cases and results. Finally, Section 10 concludes the article outlining future research directions.

2 Related work

In this section we consider prior work related to the three main challenges that this work addresses: (a) The use of reinforcement learning techniques in resolving congestion problems and in particular to solving DCB problems in the ATM domain, (b) scalable deep reinforcement learning methods, and (d) explainable deep reinforcement learning. We succinctly present the advantages and limitations of previous efforts, motivating our choices towards our contributions.

2.1 Reinforcement learning for solving congestion problems

The potential of reinforcement learning methods (either centralized or multiagent methods) to congestion problems, other than those in the aviation domain (e.g. to urban traffic) has received much attention in the recent years, with the most challenging issue being the coordination among agents, so as the solutions to increase agents' individual payoff, in conjunction to increasing the whole system utility. Towards this target there are several proposals, among which the use of coordination graphs [3], where agents coordinate their actions only with those whose tasks somehow interact.

The use of coordination graphs, where agents connected in pairs have to decide on joint policies, connects the computation of joint policies to computing equilibria in Markov games between interacting agents. Towards this goal, early studies (e.g. [20–22]), have shown that Q-learners are competent to learners using for instance WoLF [23], Fictitious Play [24], Highest Cumulative Reward -based [25] models. Based on these conclusions, the work in [26] proposes social Q-learning methods, according to which agents interact with their acquaintances, considering their tasks in their social contexts, w.r.t. operational constraints. This happens in contrast to other approaches where agents learn by iteratively interacting with a single opponent from the population [21, 27], or by playing repeatedly with randomly chosen neighbours [28].

Our work goes beyond state of the art methods in resolving congestion problems in any domain, where either a centralized agent learns a global policy, or multiple independent (i.e. non-interacting) Q-learners learn their policies, considering the other agents as part of their environment. Exceptions to this is the method proposed in [3], where instead of collaborative reinforcement learning methods, the max-plus algorithm has been used, and the method proposed in [29], where a model for incorporating multiple deep reinforcement learners is proposed.

2.2 Resolving the DCB problem

A comprehensive review of mathematical modelling and various formulations of demand-capacity imbalance problem is presented in [30]. This work reviews methods addressing congestions due to excess of the airport arrival and departure capacities, or of the airspace sector capacity. While most of early work refers to the simplest models, which do not consider airspace sectors, a category of methods addressing the Air Traffic Flow Management Problem attempts to solve real situations, also considering the airspace sector capacity. Additionally, while ground and en-route delays are important measures studied towards resolving congestions, methods addressing the Air Traffic Flow Management Rerouting Problem consider also the case where the flights can be diverted to alternative routes. Authors in [31] focus on the importance of adding rerouting as an additional mechanism, although at a smaller scale, as far as the number of flights, origin-destination pairs and air space segments are concerned. As the authors of [30] point out, the problem becomes more realistic when changes in capacity are considered, which has led to incorporating stochastic methodologies for possible unforeseen changes. These methods focus mostly on the tactical phase of operations, rather on the pre-tactical.

Other efforts has shown the importance and potential of multiagent reinforcement learning methods to address congestion problems in Air Traffic Management at the tactical level [1, 4, 32–34]. This provides a shift from the current ATM paradigm, which rely on a centralized, hierarchical process, where decisions are based on flow projections ranging from one to six hours, resulting to slow reactions to developing conditions, potentially causing minor local delays to cascade into large regional congestion.

In contrast to efforts on predicting the demand and delays in a data-driven manner [35, 36], as well as to the rich literature and various formulations of demand-capacity balance problem in the context of the Air Traffic Flow Management problem (among which those mentioned above) at the tactical phase (i.e. during operation), our work considers solving the DCB problem at the pre-tactical phase. Thus, we do neither aim to predict regulations under certain conditions based on historical data, nor we aim to resolve conflicts during the tactical phase: We rather aim to resolve the DCB problem at the pre-tactical phase considering the problem in an airspace comprising multiple sectors (e.g. the airspace of Spain) and for long time horizons, minimizing the regulations and measures applied to flights.

This is also the case in our previous work, where we introduce a new paradigm for solving DCB problems using an agent-based approach and where we show the potential of reinforcement learning to provide qualitative solutions compared to what happens today, however with a delta to the realistic setting we consider in our current work. Specifically, our previous work explores tabular Q-learning methods, following the individual learners' approach either in a flat or in a hierarchical way, while also comparing these approaches to collaborative tabular Q-learning methods exploiting a dynamic coordination graph [16–18].

Here we extend these works by following the DQN deep Q-learning method, being able to be trained efficiently for a large number of agents, in a high dimensional state space, incorporating numerous state features. In relation to tabular hierarchical approaches, as authors in [37] have shown, DQN offers spatio-temporal state abstractions, since the features learned aggregate the state space in a hierarchical fashion in spatial and temporal dimensions.

2.3 Scalable deep reinforcement learning

Scalability regarding the size of the population of Deep Reinforcement Learning agents is a major and well documented issue [38–42] that becomes apparent in many real-life problems. A large agent population could mean anything from hundreds up to several thousands of agents, as is in our use case.

There are several factors which affect training scalability. Firstly, the training paradigm adopted, i.e., agents may train independent, centralized or shared models. Secondly, the types of models learnt following any paradigm, mainly divided into two categories, the policy and the actor-critic models. Thirdly, assumptions regarding agents homogeneity are crucial, as agents may be heterogeneous, homogeneous, or even interchangeable. In addition, effectiveness of communication can play a central role. Finally, decomposing rewards among agents is a relevant issue, affecting both scalability and the quality of joint policies.

A technique which is vital in Deep MARL is centralized training and decentralized execution (CTDE): In order to alleviate problems of inefficiency and instability in learning, CTDE employs a form of centralized training, thus exploiting information that is available during training but unavailable during execution. Parameter sharing [43] is the extreme case of CTDE, as it learns a single policy shared by many agents. The main idea is that this single policy should be able to adequately describe the behavior of different agents with the same goals. The resulting policy can be robust, given the fact that it has been trained with samples that potentially belong to different parts of the state space, explored by different agents. This technique provides excellent scalability, as the number of trained models does not change as the agent population increases.

CommNet [44] has major limitations regarding scalability. Firstly, all agents use the same centralized network, which can become problematic if the agent population contains thousands of agents. In addition, in CommNet all existing agents have to communicate with everyone while the mean of messages are calculated, therefore assuming that all agents are of equal importance. This could result in significant noise in the communication channel, as an agent could receive a communication vector consisting of the average observations of irrelevant agents.

Another interesting approach, which focuses on learning how to communicate while learning a policy, is Bayesian action decoder (BAD) [45]. The method introduces the public belief MDP, and utilizes it to approximate a Bayesian update to obtain a public belief, which conditions on the actions taken by all agents in the environment. It achieves state of the art results in the game of Hanabi [46], a fully cooperative card game in which communication between players is paramount. Scaling this method to populations of agents in the order of magnitude of thousands is a non-trivial task. Separate agent neighborhoods would need separate public beliefs, in order to avoid noise in the communication scheme. In addition, as shown in [47], Policy Gradient methods present significant sample efficiency deterioration when applied to problems with large agent populations.

A method with scalability potential is BiCNet [38]. The major drawback is that, similarly to CommNet, all agents communicate with everyone. Later works [48] show that the ability of BiCNet to learn effective policies is reduced as the number of the agents increases. This deterioration of effectiveness is attributed to the lack of a mechanism capable of capturing the importance of information from different agents.

Mean Field MARL [47] proposes two algorithms, a Q-Learning and an Actor-Critic variant, with the explicit aim to improve MARL scalability. The main idea is to calculate the mean action of an agent's neighborhood, assuming that each agent interacts with a virtual mean agent. Both algorithms are empirically shown to scale up to 1000 agents and should in theory be applicable in our use case of around 7000 agents. Despite the scalability of both algorithms, their main drawback lies in the coordination scheme, as averaging neighborhoods can result lackluster cooperation [49].

Graph Convolutional Reinforcement Learning (DGN) [49] proposes a MAS framework in which agents are connected in a graph where each agent is a node and edges exist between neighbors. Communication is allowed only between neighbors, in order to minimize inefficiency. The method was compared against well-known algorithms like CommNet [44] and MFQ [47] and is shown to achieve very competitive results in environments with up to 140 agents. As shown in the experimental section of [47] approaches that produce qualitative results in settings with a few hundred agents may significantly deteriorate in sample efficiency with thousands of agents.

K.Lin et al. in [50] proposes two distinct methods which, similarly to our use case, aim to resolve large scale demand-capacity imbalance problems in the air traffic management domain. The environment simulates a population of approx. 5000 homogeneous agents. An assumption that both methods make is that agents have the same action values. In practice, this assumption means that the agents are not simply homogeneous, but interchangeable. Closely related works, with scalability in mind, are those in [51–53]. They achieve extreme scalability and provide experiments with up to 8000 agents. A fundamental idea underlining the work described in these papers and a vital aspect to achieve scalability, is exploiting the count of agents in state s taking action a. This count, as well as other more complex measures based on this, serve a statistical basis for training, eliminating the need to collect trajectory samples from every agent, so the resulting policy is dependent on count-based observations. Similarly to [50], this approach assumes that the agents are interchangeable.

To address scalability and in order to alleviate problems caused by the non-stationarity of independent learners, we do employ centralized training, exploiting information that is available during training but often unavailable during execution. In so doing we are using the centralised training and decentralized execution (CTDE) technique, aiming to learn a single policy shared by many agents. Our hypothesis is that the resulting policy can be generalizable and robust, given the fact that it is trained with samples that potentially belong to different parts of the state space, explored by different agents.

In addition to CTDE, and in contrast to other works considering explicit communication between agents, we have made a conscious design choice to not include explicit inter-agent communication, in order to avoid the communication cost, which can become significant when working with population sizes as big as 6000 agents or bigger. Of course, there is implicit communication among agents due to the fact that all agents share the same network parameters and explore the same state-action space, while we do consider in an explicit manner perturbations caused by the actions of other agents acting simultaneously.

2.4 Explainable deep reinforcement learning

DRL models, are inherently hard to interpret. We need methods that extract explanations (i.e. interpret these methods) in either a local or in a global way: By "local" we mean providing explanations for decisions taken in fine granularity and scale, while by providing "global" explanations we consider providing information of the entire logic (means and/or ends) towards achieving goals.

To interpret DRL methods without using interpretable models as DRL components, a method may use samples provided by the DRL method, potentially exploiting further information from DRL models. This results into two paradigms: The mimicking and the distillation paradigms.

For DRL methods, the distinguishing line between the distillation and the mimicking processes is not that clear in the literature. For instance, in reference [54], the DRL model provides samples, each comprising an observation sequence and a vector of state-action Q-values. We can say that the main difference between the two paradigms concerns the input provided to the interpretation process: While the distillation process distils the knowledge acquired by the trained DRL agent by exploiting directly information from any of the constituent DRL models, the mimicking process monitors the interaction of the reinforcement learning agent with the environment and gathers interaction samples, recording agent decisions, state transitions and rewards that the agent got.

Specifically, in the ATM domain, explainability of AI algorithms has not been extensively studied, as also mentioned in [55]. Going one step further, authors in [55] identify three major pieces of information that stakeholders in ATM should be provided by AI modules when these are induced within the complex ATM system: (a) Descriptive explanations, as the system should be able to provide the detailed description of the situation, motivating the actions to be taken; (b) Predictive, by providing information of the consequences of actions to be taken; and (c) Prescriptive, by being able to propose the appropriate actions and options, along with an appropriate explanations, so as stakeholders to decide on the next course of actions.

In this work we follow the mimicking paradigm for explaining DRL decisions. We use a state of the art method for training interpretable models, not used before for this purpose: Stochastic Gradient Trees. These have been shown to outperform other types of tree models that have been used to interpret DRL models. In so doing, we strive to cover all three requirements for explainability in the ATM domain. We propose a pipeline that is able to describe situations under which decisions have been taken, provide arguments for the proposed actions, visualize their consequences, and give alternative solutions in the form of imposing different types of regulations.

3 Problem specification

3.1 Demand-capacity balance: Definition of terms

While the trajectory becomes the cornerstone upon which all the ATM capabilities will rely on[1], flight trajectories cannot be considered in isolation: Intertwined operational aspects lead to inefficiencies to trajectory planning and introduce factors of uncertainty to trajectory execution. Accounting for network effects and their implications on the joint execution of individual flights, requires considering interactions among trajectories, and thus, dynamic operational conditions that influence any flight.

Being able to devise methods that capture aspects of that complexity and take the relevant information into account, would greatly improve planning and decision-making abilities in the ATM domain.

Towards this goal, our specific aim is to automate the resolution of DCB problems, by deciding the measures that need to be imposed to planned flight trajectories (flight plans) before the actual operations, considering ATM system dynamics and network effects due to interactions among trajectories.

More specifically, the DCB problem (or DCB process) considers two important types of objects in the ATM system: *aircraft trajectories* which in our case are *flight plans*, and *airspace sectors*. Sectors are air volumes that divide the airspace.

Airspace sectorizations determine the active (open) sectors at any time instance. Only one sectorization can be active at a time. Airspace sectorization changes during the day, given different operational conditions and needs. This happens transparently for flights.

The *capacity* of a sector is of utmost importance in DCB: this quantity determines the maximum number of flights that can enter a sector during any time period of specific duration (typically, 60').

There are different types of measures to monitor the demand evolution, with the most common ones being the

[1] Due to different initiatives, notably SESAR in Europe and Next Gen in the US

Hourly Entry Count and the *Occupancy Count*. In this work we consider *Hourly Entry Count*, as this is the one used by the Network Manager (NM) at the pre-tactical phase.

The *Hourly Entry Count (HEC)* for a given sector is defined as the number of flights entering the sector during a time period, referred to as an *Entry Counting Period* (or simply, *counting period*). HEC measures demand providing a "snapshot" of the entry traffic in a sector, taken at every step value along a counting period of fixed duration: The step value defines the time difference between two consecutive counting periods. The duration of the counting period is equal to the duration of the period used for defining capacity. For example, for a 20 minutes step value and a 60 minutes counting period duration, entry counts correspond to pictures taken every 20 minutes, over 60 minutes. Figure 1 shows the evolution of demand in sector LECPDEO during a day using a step of 20 minutes, over a duration of 60 minutes.

Let us consider a finite set of air sectors $\mathbf{R}$=$\{R_1, R_2, ...\}$ segregating the airspace. Demand-capacity imbalances occur when $D_{R,p} > C_R$, where p is a counting period of duration d, $D_{R,p}$ is the demand for the *active* sector R during p, and C_R is the capacity of that sector, defined for any such period.

Aircraft trajectories are defined to be series of spatio-temporal points ($long_i$, lat_i, alt_i, t_i), where each point denotes the longitude, latitude and altitude of the aircraft at a specific time point t_i. Casting trajectories into a DCB resolution setting, these are specified to be time series of events specifying the time points at which the trajectory enters and exists each sector crossed. Therefore, a trajectory T is specified to be of the form:

$$T = \{(R_1, entry_1, exit_1)....(R_m, entry_m, exit_m)\}, \tag{1}$$

where $R_l, l = 1, ..., m$, is an active airspace sector and *entry/exit* are time points of entering/exiting that sector. This information per trajectory suffices to monitor demand evolution for the active sectors, in any counting period p.

Given a set of such trajectories, the demand in sector R in counting period p is $D_{R,p} = |\mathbf{T}_{R,p}|$, i.e., the number of trajectories in $\mathbf{T}_{R,p}$, where:

$\mathbf{T}_{R,p} = \{T \in \mathbf{T} | T = (\ldots, (R, entry_t, exit_t), \ldots)$, *and the temporal interval* $[entry_t, exit_t]$ *overlaps with counting period* p $\}$.

Trajectories requiring the use of a sector R at the same counting period p, i.e. trajectories in $\mathbf{T}_{R,p}$ are *interacting trajectories* for p and R.

3.2 Operational context

We assume a trajectory-based operations environment, with an enhanced accuracy of pre-tactical flight information – provided by airlines' flight plans. This operational environment is close to the one existing today, but it requires airlines to specify their flight plans during the pre-tactical phase, which, together with airspace operational constraints allow the detection of hotspots.

In this operational context we consider an agent A_i to be the aircraft performing a planned flight trajectory, in a specific date and time. Thus, we consider that agents and trajectories coincide, and we may interchangeably speak of agents A_i, trajectories T_i, flights, or agents A_i executing trajectories T_i. Agents, in the general case can have own interests, preferences and restrictions, and take autonomous decisions on resolving hotspots.

3.3 Resolution of demand-capacity imbalances

To resolve DCB problems at the pre-tactical stage, airspace users have several degrees of freedom: They may either (a) change the trajectory of flights to cross sectors other than the congested ones, devising alternative flight plans via re-routing, or (b) change the entry and exit time for each of the crossed sectors by imposing a ground delay, i.e. delaying take-off. In this paper we consider DCB resolution measures implying (a) vertical re-routing due to *level capping*: i.e. changing the vertical profile of the flight plan, restricting flights to climb to a certain level so as to avoid congested sectors at high levels; and (b) changes in the schedule of crossing sectors by means of *ground delays*: i.e., shifting the whole trajectory by a specific amount of time.

Overall, agents decide on the measures that they will apply so as to execute their trajectories jointly, resolving DCB problems.

Specifically, the objectives of the agents are to

- Resolve demand-capacity imbalances, providing a solution with a minimized number of hotspots, which shall be tolerated or be resolved at the tactical phase; in conjunction to
- minimize the average delay per flight (ratio of total delay to the number of flights) without imposing re-routings that increase the operational costs.

In doing so, the system must

- distribute delays to flights without penalising a small number of them, and
- utilise efficiently the airspace, distributing demand to sectors evenly in all counting periods during trajectories' execution period.

3.4 A multi-agent system perspective

To resolve a hotspot occurring in counting period p and sector R, a subset of interacting trajectories in $\mathbf{T}_{R,p}$ must be re-routed or delayed.

Imposing demand measures to trajectories may cause the emergence of hotspots to subsequent time periods for the same and/or other sectors, resulting to a dynamic setting for any of the agents: the sets of interacting trajectories do change unpredictably for the individual agents, due to agents' joint decisions and changes in sectorization. Furthermore, while the decision of one of the interacting trajectories directly affects the others, a trajectory may *indirectly* affect any other trajectory, due to ATM "network effects".

The *society* of agents $(\mathbf{A}, \mathbf{E^t})$ can be modelled as a dynamic *coordination graph* [56] with one vertex per agent A_i in $\mathbf{A}$. An edge (A_i, A_j) in $\mathbf{E^t}$ connects agents with interacting trajectories in $\mathbf{T}$, at time t. The set of edges are dynamically updated when the set of interacting trajectories changes.

$\mathbf{N^t}(A_i)$ denotes the *neighbourhood* of agent A_i in the society, i.e. the set of agents interacting with agent A_i at time instant t, including itself.

The *ground delay* options available in the inventory of any agent A_i may differ between agents: These are in $\mathbf{D_i} = \{0, 1, 2, ..., MaxDelay_i\}$, and each option specifies minutes of delay. Regarding level capping, the options of agent A_i are in a set $\mathbf{FP_i}$, specifying flight plans. Overall, the set of options available for each agent A_i are in $\mathbf{FP_i} \times \mathbf{D_i}$.

We consider that options for delays and re-routing may be ordered by the preference of agent A_i, according to the functions $p_i^{flightPlans} : \mathbf{FP_i} \rightarrow \mathbb{R}$ and $p_i^{delays} : \mathbf{D_i} \rightarrow \mathbb{R}$. We do not assume that agents in $\mathbf{A} - \{A_i\}$ have any information about these functions. This represents the situation where airlines set own options and preferences for alternative flight plans, even in different own flights, depending on operational circumstances, goals and constraints. We expect that the order of preferences on delays should be decreasing from 0 to $MaxDelay_i$, although, with a different pace/degree for different agents. The preferences on flight plans depends on the cost-effectiveness of the plan. For instance, flight plans with abrupt and/or frequent changes in flight levels are less favorable than plans with a smooth vertical profile. We address this issue in subsequent sections.

3.5 Problem statement (Multi-agent DCB problem resolution)

Considering any pair of interacting agents A_i and A_j in the society $(\mathbf{A}, \mathbf{E^t})$, with A_j in $\mathbf{N^t}(A_i) - \{A_i\}$, they must select among the sets of available options in $\mathbf{FP_i} \times \mathbf{D_i}$ and $\mathbf{FP_j} \times \mathbf{D_j}$ respectively, so as to increase their expected payoff w.r.t. their objectives and preferences on delays and flight plans.

This problem specification emphasises on the following problem aspects:

- Agents need to coordinate their decisions to execute their trajectories jointly with others, considering traffic and network effects, w.r.t. their objectives, preferences and operational constraints;
- The setting is highly dynamic given that the agents' society, the occurring hotspots, and the sectorization change unpredictably for agents.
- Agents need to jointly explore and discover how different combinations of level-capping and delay measures affect the joint performance of their trajectories, given that the ways different trajectories do interact cannot be predicted;
- Agents' preferences and constraints on the options available may vary depending on the trajectory performed, and are kept private;
- There can be multiple and interdependent hotspots that occur and agents aim to resolve all of them.

4 Multi-agent DCB policy search problem formulation

According to the problem specification stated above we formulate the multi-agent DCB policy search problem as an MDP comprising the following constituents:

- A set of time instances $t = t_0, t_1, t_2, t_3, ..., t_{max}$, s.t. $t_{max} - t_0 = \mathbf{H}$ and $t_{i+1} - t_i = \Delta t, i = 0, ...max - 1$, where $\mathbf{H}$ is a *time horizon* and Δt a *timestep*.
- The dynamic *society* of agents $(\mathbf{A}, \mathbf{E^t})$ at time t, as described above.
- A set of *agent states*: A local state per agent A_i at time t, denoted by s_i^t, comprises state variables that correspond to (a) the delay imposed to the trajectory T_i executed by A_i, ranging to $\mathbf{D_i} = 0, \ldots, MaxDelay_i$, (b) the number of hotspots in which A_i is involved in during $\mathbf{H}$, (c) the sector IDs that it crosses, (d) the minutes A_i it remains within each sector it crosses (e) counting periods in which A_i participates in hotspots and the corresponding sectors, and (f) the minute of the day that A_i takes-off, given $\mathbf{D_i}$.

  The joint state s_{Ag}^t of a set of agents Ag at time t is the tuple of all agents in Ag local states. A *global (joint) state* $\mathbf{s}_t$ at time t is the tuple of all agents' local states. The set of all *joint states* for a subset Ag of agents is denoted $\mathbf{State}_{Ag}$, and the set of *joint society states* is denoted by $\mathbf{State}$.
- The set of *agent actions*: A local action (decided measure) for agent A_i at time t, denoted by a_i^t is a choice of an option in $\mathbf{FP_i} \times \mathbf{D_i}$ (i.e., the agent acts to impose a demand measure). This action results from agent's decision on changing the flight plan: Given a flight plan, at each time point until take-off the agent

has to take a decision on the additional minutes of delay up to that time point. So, given the delay D_i^t decided up to a time point t the agent may add to its total delay $d_i^t \in \{0, 1, ..., \Delta t\}$, minutes, given that $D_i^t + d_i^t \leq MaxDelay_i$. Similarly, in the case of level capping solutions, the choice is in $\mathbf{FP_i}$, and it is a variation of trajectory T_i produced by imposing level capping measures.

The *joint action* of a subset of agents Ag of $\mathbf{A}$ at time t (e.g. of $\mathbf{N}^t(Ai)$), is a tuple of local actions decided by agents, denoted by $\mathbf{a}_{Ag}^t$ (e.g. $\mathbf{a}_{\mathbf{N}^t(A_i)}^t$). The joint action for all agents $\mathbf{A}$ at any time instant t is denoted $\mathbf{a}_t$. The set of all *joint actions* for any subset Ag of $\mathbf{A}$ is denoted $\mathbf{Action}_{Ag}$, and the set of *joint society actions* is denoted by **Action**.

- The *state transition function Tr* gives the transition to the joint state $\mathbf{s}^{t+1}$ based on the joint action $\mathbf{a}^t$ taken in joint state $\mathbf{s}^t$. Formally:

$$Tr : \mathbf{State} \times \mathbf{Action} \rightarrow \mathbf{State}, \quad (2)$$

It must be noticed that the state transition per agent is stochastic, given that no agent has a global view of the society, of the decisions of others, and/or of changing sectorization, while its *neighbourhood* gets updated. Thus, no agent can predict how the joint state can be affected in the next timestep. Thus, from the point of view of agent A_i this transition function is actually:

$$Tr : \mathbf{State}_{A_i} \times \mathbf{Action}_{A_i} \times \mathbf{State}_{A_i} \rightarrow [0, 1], \quad (3)$$

denoting the transition probability $p(s_i^{t+1}|s_i^t, a_i^t)$.

- The *local reward* of an agent A_i, denoted Rwd_i, is the reward that the agent gets in a specific state at time t. The *joint reward*, denoted by Rwd_{Ag} for a set of agents Ag specifies the reward received by agents in Ag by executing their trajectories according to their *joint action*, in a *joint state*. Further details on the reward function are provided in Section 6.
- A *(local) policy* of an agent A_i is a function π_i : $\mathbf{State}_{A_i} \rightarrow \mathbf{Action}_{A_i}$ that returns an action for any given local state.

It must be noted that each agent, according to its decided action, either (a) chooses a flight plan by re-routing, without imposing any ground delay to its flight, (b) keeps the original flight plan and chooses the ground delay to be imposed, or (c) changes the flight plan by means of re-routing and imposes a ground delay. We consider all these alternatives separately, thus resulting to different types of solutions to a DCB problem. In the later case where agents choose re-routing and delay, we consider it as a two stage approach: Agents choose their flight plan from those in $\mathbf{FP_i}$ and given their chosen flight plan they decide on the ground delay from options available in $\mathbf{D_i}$. An alternative way to choose an action among the available options at the pre-tactical stage of operations would be to choose conjunctively a flight plan and the ground delay. However, this unnecessarily explodes the size of the state-action space.

The objective for any agent $A_i$in the society is to find an optimal policy π_i^* that maximizes the expected discounted future return for each state s_i^t, given the initial state s_i^1.

$$V_i^*(s) = max_{\pi_i^*} E\left[\sum_{t=1}^{\infty} \gamma^{t-1} Rwd_i(s_i^t, \pi_i^*(s_i^t))|s = s_i^1\right] \quad (4)$$

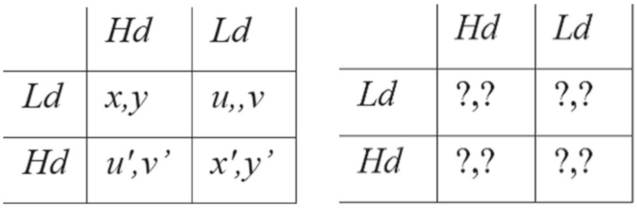
The discount factor γ ranges in [0,1].

This model assumes the Markov property, assuming also that rewards and transition probabilities are independent of time. Thus, the state next to state s given a (joint) action is denoted by *s'* and it is independent of time. Subsequently, subscripts and superscripts are avoided in cases where it is clear where a state or action refers to.

4.1 DCB resolution as a Markov game

The problem can be considered as a specific instance of the multi-agent coordination problem specified in [26], where each agent has several options to execute a single task (trajectory in this case), while agents' tasks must jointly satisfy operational constraints.

In doing so, the resolution of any DCB problem can be formulated as a Markov game: Let us consider a society with two agents executing interacting trajectories and causing a hotspot. Let us also assume that each agent, has two (e.g. level capping) options: a low (Ld) and a high (Hd) one. Assuming, without loss of generality, that one of the agents should choose Hd to resolve the DCB problem (otherwise either there is no hotspot, or the hotspot will re-occur later in time), agents are assumed to play a game of the form shown in Fig. 2a: All entries in this matrix are different than zero; *x, x, y, y'* can be considered positive integers (in case the hotspot is resolved); *u, v, u', v'* can be negative integers (in case agents do participate in a hotspot). As it can be noticed, this can be a coordination game, with two Nash equilibria, namely the joint options providing payoffs *(x,*



	Hd	*Ld*
Ld	*x,y*	*u,,v*
Hd	*u',v'*	*x',y'*

(a)

	Hd	*Ld*
Ld	?,?	?,?
Hd	?,?	?,?

(b)

Fig. 2 Payoff matrices for 2X2 games

y) or *(x', y')*. However, this is not necessarily a symmetric game, considering that the payoff incorporates agents' preferences. The game can be extended to multiple options and/or multiple agents executing interacting trajectories.

Given that the information concerning the effects of agents' joint decision is not known to any agent in the society, and given that agents do not know about the payoffs of other agents when choosing specific delay options, agents need to learn about the structure of the game to be played, and they have to coordinate with others to play the game. The information that an agent has about a 2 × 2 game is as shown in Fig. 2b. Question marks indicate the missing information: For instance, none of the two agents knows whether a decision is effective in resolving hotspots, nor the payoffs from joint decisions (e.g. since new hotspots may emerge as a result of agents' joint decision). Our goal is agents in the society to converge to a joint decision, so as to resolve hotspots that occur jointly with all society members.

5 Explainability desiderata to promote trust

As pointed out in the introductory section, we aim at introducing automation and promoting system trustworthiness, in the sense that operators should be comfortable relinquishing control to the system, given appropriate explanations on system's decision making. Therefore, the system must be able to provide explanations of its decisions. Challenges towards addressing this objective include: (a) The large number of agents participating, (b) the fact that - as pointed out in the MDP formulation - agents decide every Δt time points on the additional minutes of delay up to $MaxDelay$, or until they reach the take-off time point, (c) the inherent complexity of the system.

Therefore, given humans' cognitive limitations, it is challenging to provide a global understanding of agents' policy, to inspect the contribution/effect of every single agent decision to the overall joint solution; or even to understand patterns of behaviour that emerge from the decisions of interacting agents.

However, operators need (a) to get a qualitative understanding of the system, (b) get a justification of agents' decisions at the appropriate level of granularity and scale, and (c) understand the importance of state features in agents' decision making.

Qualitative understanding expresses how humans understand the policies of agents, their preferences, abilities and objectives, being able to inspect the rationale and effects for agent's courses of decisions, as the environment evolves within a state space.

This is equivalent to supporting humans to model how agents respond in certain circumstances w.r.t their abilities, preferences and objectives, without considering low level algorithmic details and numerical assessments/computations that the system makes.

Regarding the second requirement, while the *granularity* concerns the level of detail regarding agents' behaviour, and it may refer to the specific decisions taken at any timestep, or to the mode of behaviour or skill exercised by any agent within a time period, *scale* refers to the temporal or to the spatial extent in which decisions are considered. For instance, one may consider agents' decisions at any timestep (fine level of granularity) for a time horizon of X hours and only within specific space compartments (temporal and spatial scale).

Finally, understanding the importance of state features, allows humans to understand where do agents "pay attention" while taking any decision.

Towards this goal, in this paper we do explore the use of appropriate visualizations and visual analytic tools that allow operators to explore the system behaviour at various temporal, spatial and societal scales, while providing interpretations of individual agent decisions at low levels of granularity.

The main point here is that in large scale and complex systems human operators need time to explore the solutions proposed by the system, to explore agents' interactions and consequences of decisions. In so doing, trust to the system is developed while operators are acquainted with the system and while they review the rationale behind intertwined agents' decisions in multiple cases. In our case, automation is introduced at a pre-tactical phase, so indeed, humans have the time to delve into the details of solutions.

6 Multi-agent deep Q learning

6.1 The deep Q learning method

To resolve problems in high dimensional and large state-action spaces, the deep Q-network [57] method successfully combined Reinforcement Learning (RL) with neural networks (NNs). Deep Q-Network (DQN) uses a NN ("online network") to approximate the optimal action-state Q function

$$Q^{*}(s,a) = max\mathbb{E}[Rwd_{t} + \gamma Rwd_{t+1} + \gamma^{2} Rwd_{t+2} + ...|s_{t} = s, a_{t} = a, \pi^{*}] \tag{5}$$

which, is the maximum expected sum of rewards discounted by γ at each timestep t, achievable by policy π^{*}, after state s and taking action a.

It is well established that the combination of RL with nonlinear function approximators such as NNs, can be unstable [58]. This instability has several causes: the correlations present in the sequence of observations, the fact that small updates to Q values may significantly change the policy - and therefore change the data distribution, and the correlations between the action-values and the target values.

As shown in Fig. 3, two vital elements of DQN that address these issues, are the target network and the experience replay memory. The target network mitigates the effect of constantly moving update targets, by incorporating a second network from which the update targets are sampled. This network is periodically updated with the weights of the online network. The addition of a uniform experience replay memory de-correlates the samples collected during rollouts, by randomizing over the data, thereby smoothing over changes in the data distribution. During learning, the method applies Q-learning updates on samples (or minibatches) of experience drawn uniformly at random from the pool of samples stored. The Q-learning update at iteration i uses the following loss function:

$$L_i(\theta_i) = \mathbb{E}_{(s,a,Rwd,s')\sim U(D)}[(Rwd + \gamma max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i))^2] \tag{6}$$

In which γ is the discount factor, θ_i are the parameters of the online network at iteration i and θ_i^- are the target network parameters at iteration i.

Regarding the number of agents participating in a typical DCB problem setting (typically, more than 6000), scalability is an important challenge. While the straightforward way to train agents towards a joint policy, is to train independent policies in parallel, one for each agent, following the independent learners paradigm, this approach can produce results in low-dimensional problems with only few agents: In real-world, large scale settings is inefficient and unstable. As already pointed out in Section 2, to alleviate problems caused by the non-stationarity of independent learners and in order to address scalability, we do employ some form of centralized training, thus exploiting information that is available during training but often unavailable during execution. A technique that has many variants and is vital in understanding Deep MARL, while tackling scalability issues, is the centralised training and decentralized execution (CTDE) technique. Following the parameter sharing [43] paradigm, we aim to learn a single, generalizable and robust policy shared by many agents. The main idea is that this single policy should be able to adequately describe the behaviour of different agents with the same goals.

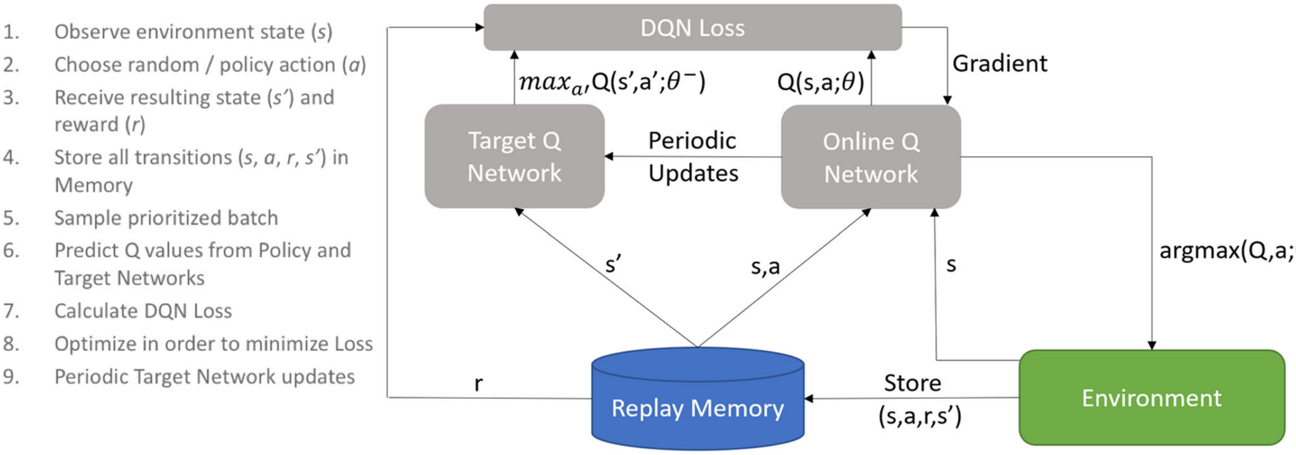
At every step of the environment, transition samples are collected from every agent and stored in the experience replay memory. These samples contain the state of the agent s, the action decided a, the resulting state s' and reward Rwd (i.e., (s, a, Rwd, s')).

Furthermore, we incorporated three enhancements of DQN in order to improve stabilization of the method and quality of results.

The first extension is Double DQN [59]. Double Q Learning was originally introduced in [60] and aims at addressing the problem of overestimating action values: A phenomenon inherent to the DQN method. This addition to the original method is considered to be standard practice and is particularly useful in the multi-agent domain, where non-stationarity is a common phenomenon. Originally, the idea behind this method is to utilize two independent tabular approximations of the Q function during training, where each Q function is updated with targets produced from the other Q function. Specifically, in its tabular version, given a Q function approximation A and a second one B, it uses the following formula to update A. The update for B is symmetrical.

$$Q^A(s,a) = Q^A(s,a) + \alpha(Rwd + \gamma Q^B(s',a^*) - Q^A(s,a)) \tag{7}$$

Double DQN transfers this approach in the Deep RL setting, by exploiting the already existing second approximator in the form of the target network. The online policy network is used to choose the best next action a',



1. Observe environment state (*s*)
2. Choose random / policy action (*a*)
3. Receive resulting state (*s'*) and reward (*r*)
4. Store all transitions (*s, a, r, s'*) in Memory
5. Sample prioritized batch
6. Predict Q values from Policy and Target Networks
7. Calculate DQN Loss
8. Optimize in order to minimize Loss
9. Periodic Target Network updates

Fig. 3 DQN architecture

but its Q value is evaluated by the target network. Thus the original Q-learning target

$$Y_t{}^{Q} = Rwd_{t+1} + \gamma Q(s_{t+1}, argmax_a Q(s_{t+1}, a; \theta_t); \theta_t) \tag{8}$$

becomes

$$Y_t{}^{QDouble} = Rwd_{t+1} + \gamma Q(s_{t+1}, argmax_a Q(s_{t+1}, a; \theta_t); \theta_t^-) \tag{9}$$

The second extension is the prioritized experience replay [61]. In the original approach, experience transitions are uniformly sampled from a replay memory. This approach ignores the significance of samples, replaying them at the same frequency that they were originally observed in the environment. Prioritized experience replay enforces a priority over the samples, aiming to replay important transitions more frequently, and subsequently improve learning efficiency. In particular, the method proposes to assign higher priority to transitions with high expected learning progress, as measured by the magnitude of their temporal-difference (TD) error. Specifically, a stochastic sampling method interpolates between pure greedy prioritization and uniform random sampling, while guaranteeing a non-zero probability even for the lowest-priority transition. The probability of sampling transition i as

$$P(i) = \frac{p_i^{\eta}}{\sum_k p_k^{\eta}} \tag{10}$$

where, $p_i > 0$ is the priority of transition i and the exponent η determines how much prioritization is used. $p_i = |\delta_i| + \beta$, where δ_i is the TD error of sample i and β a small positive constant.

In our case, preliminary results showed that prioritized experience replay was essential. Uniform experience replay failed to obtain the optimal solution even in simple DCB cases with few (e.g. 5) agents, by resolving all hotspots, but assigning unnecessary demand measures: The combination of optimal measures that resolve all hotspots are quite rare in the experience memory and should be prioritized when encountered.

Finally, we incorporate the exploration scheme of Adversarially Guided Exploration (AGE), as proposed in [62]. As reported in [63], training on adversarial examples can lead to a more robust policy, especially against perturbations produced by the same technique. In the multi-agent domain perturbations can be produced by the agent population acting simultaneously, thus leading to the non-stationarity problem. By utilizing a form of data augmentation which produces perturbations through adversarial examples, we aim to train the policy over adversarial examples of states, which are generated when an agent acts and observes a state different than the one expected, due to other agents' actions. This results to obtaining a resilient and robust policy, against perturbations.

The AGE mechanism extends the classic ϵ-greedy exploration mechanism by adjusting the probability of sampling actions for each state s according to the adversarial state-action significance, defined as follows:

$$\zeta_{adv}^{\pi_i}(s, a) = \frac{exp(max_{a'} Q^{\pi_i}(s, a') - Q^{\pi_i}(s, a)/\epsilon)}{\sum_{a \in A} exp(max_{a'} Q^{\pi_i}(s, a') - Q^{\pi_i}(s, a)/\epsilon)} \tag{11}$$

where, $\zeta_{adv}^{\pi_i}$ measures the maximum achievable adversarial gain determined by the difference between maximum Q-value at state s and $Q^{\pi_i}(s, a)$ with respect to actions. Following the ϵ-greedy exploration paradigm, this approach chooses $argmax_a Q^{\pi_i}(s, a)$ when $\epsilon < rand()$, but in the case of $\epsilon \geq rand()$ the action is sampled according to $\zeta_{adv}^{\pi_i}$.

Subsequently, we describe how DRL has been used to compute each of the types of solutions for resolving DCB problems.

6.2 Solutions by imposing ground delays

In the first type of solutions agents can impose only delay regulations. Agents interact with a simulated environment and at each time step they can observe the resulting global state after their joint decision: It must be recalled that the environment comprises the sectorizations at different periods, capacity constraints, as well as other flights.

As already pointed out, according to our methodology, a flight is not regulated with a number of minutes at once (e.g. 14 min of delay), but by adding additional minutes of delay at every timestep, according to needs, as follows: Decisions are taken during an (simulation) episode. An episode simulates a specific scenario of a time horizon **H**=1440 minutes. A scenario comprises a set of flights flying in an airspace within **H**, and all active sector configurations during that period. Each episode comprises a series of rounds. Each round corresponds to a number of minutes of the day (mentioned as simulation timestep, and denoted by Δt in the MDP formulation). At each round each agent takes a concrete decision for *additional* delay from zero up to Δt minutes. Therefore, in case an agent (flight) has 13 minutes of delay at a time point during simulation, then there should be a number of distinct timesteps in the simulation prior to that time point, where the agent has decided to take additional minutes of delay that sum up to 13. The method takes into account the existing daily traffic, given all the initial flight plans and agents' joint decisions at each timestep. By so doing, agents are able to explore the state-action space and learn very effectively. More importantly, agents are able to justify their decisions of (not) taking additional minutes of delay at any timestep, also due to hotspots that may emerge due to combined agents' decisions.

Figure 4 shows the calculations that occur at every timestep during simulation: After agents have taken their decisions, the overall demand and the unresolved hotspots during **H** are recalculated by the simulated environment.

The reward function utilized in this type of solutions has two distinct parts. If an agent does not participate in hotspots, it receives a positive scalar value C minus the minutes of delay accumulated, divided by the maximum allowed minutes of delay. If it participates in one or more hotspots, it receives a negative reward which is equal to the ratio of (a) multiplying a positive scalar value CpM, representing the cost per minute within a hotspot, with the total duration (in minutes) of the flight in hotspots, with (b) the duration of the longest flight in the scenario.

$$Rwd_i = \begin{cases} C - \mathbf{D_i}/MaxDelay_i & hotspots_i = 0 \\ -CpM * Dur_i/MaxDur & hotspots_i > 0 \end{cases}$$

Regarding the first case of the reward function, the motivation is to provide the agent with a distinctive positive reward when it does not participate in any hotspot, i.e., what it has accomplished its main goal. In addition, we aim to minimize the minutes of delay. To regularize the negative part, delay is divided by the maximum allowed minutes of delay. Regarding the second case of the reward, the reasoning is quite similar. The negative scalar reward drives the agent to resolve all the hotspots in which it participates. It is multiplied by the total duration (minutes) that the agent is in hotspots: Larger duration indicates larger agent "contribution" in hotspots. To regularize in this case, we divide by the length of the longest flight in the scenario (which is the maximum time period that any scenario flight can participate in hotspots).

6.3 Solutions by re-routing due to level capping

In this type of solutions agents' options are restricted to the flight plans in $\mathbf{FP_i}$. This set consists of the original flight plan and revisions of this flight plan. These revised flight plans are devised so as to avoid any sector with hotspot in any counting period, via level capping measures. Our methodology starts with the production of the revised flight plans by applying level capping measures to the original flight plans, where this is possible. Thus, a revised flight plan represents a decision for applying level capping measures.

Flights eligible for level capping measures are those which take off inside the airspace of our interest, or in nearby airports. For each eligible flight, a set of flight plans with level capping measures are created and ranked according to the smoothness of the resulting planned trajectory (details about this ranking are provided in Section 9.2). This set of plans, including the original flight plan, constitute the set of flight plan options for each agent A_i, i.e. $\mathbf{FP}_i$.

After calculating the demand at the scenario initial state (please mind that this concerns all hotspots during the scenario), the method checks which flights participate in hotspots, and for each flight A_i, which plans in $\mathbf{FP}_i$ can avoid most of the hotspots. In the case where more than one hotspots can be avoided, the highest ranked flight plan that can avoid most of the hotspots in which the agent participates, is selected. Figure 5 describes the process.

Avoiding a hotspot does not mean it is resolved: Agents' joint decision may leave hotspots unresolved and they may

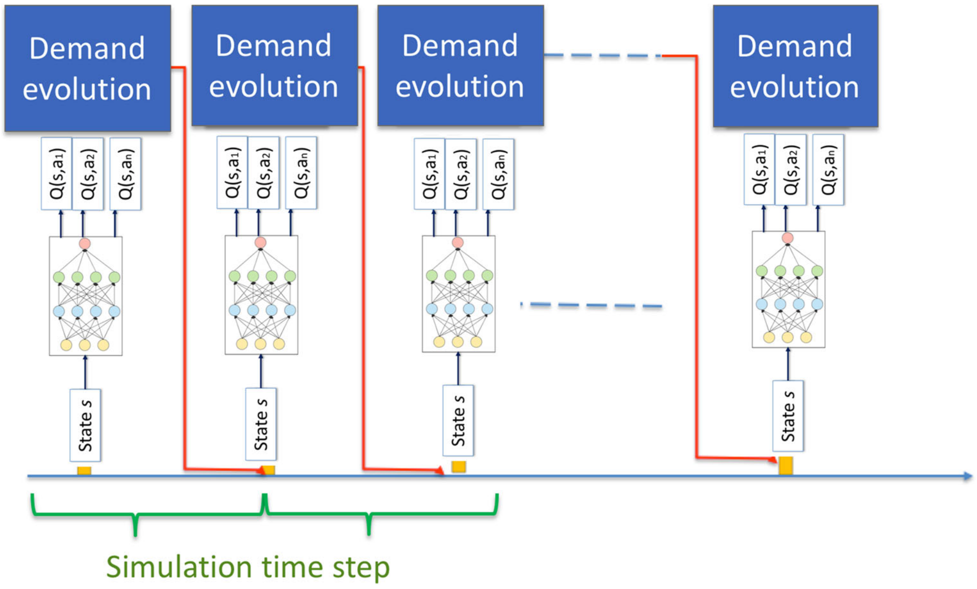


Fig. 4 Simulation rounds for delay regulations

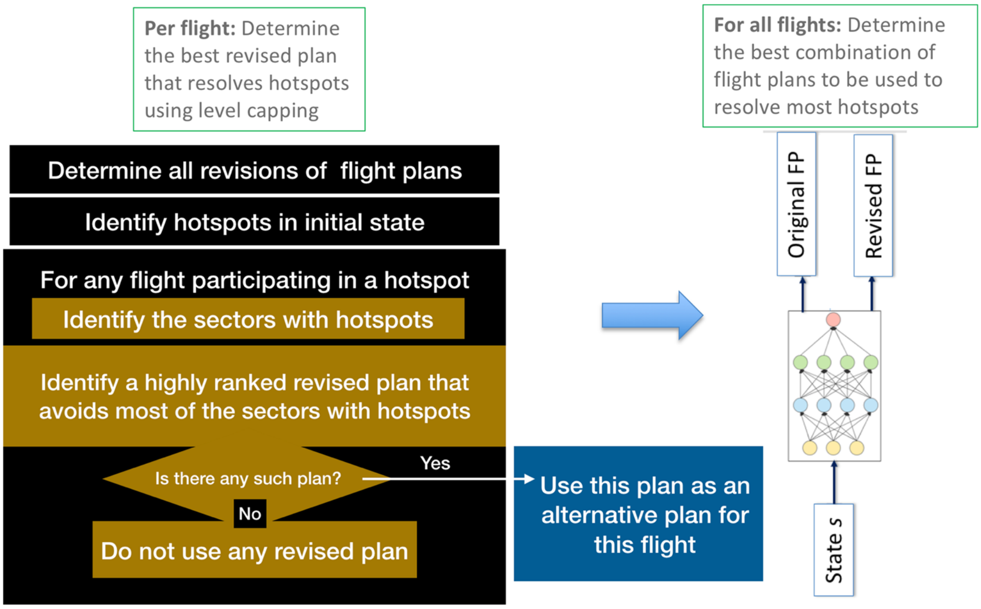

Per flight: Determine the best revised plan that resolves hotspots using level capping
For all flights: Determine the best combination of flight plans to be used to resolve most hotspots
Determine all revisions of flight plans
Identify hotspots in initial state
For any flight participating in a hotspot
Identify the sectors with hotspots
Identify a highly ranked revised plan that avoids most of the sectors with hotspots
Is there any such plan?
Yes
No
Do not use any revised plan
Use this plan as an alternative plan for this flight
Original FP
Revised FP
State s

Fig. 5 Selecting alternative flight plans due to level capping measures

create new hotspots which can be more in number and more severe in intense, than those in agents' initial state. To avoid this, the agents use the DQN algorithms to jointly decide which of the plans to follow.

In contrast to the approach described in Section 6.2 and the multi-agent MDP formulated, agents decide how to play a one-shot game, comprising the initial state and a final one where the environment transits taking into account agents' joint decision. Thus, the second and final state is the one where all agents have decided on their flight plans.

The reward function in this case has two distinct parts. If an agent participates in no hotspots it receives a positive scalar value C. If it participates in one or more hotspots it receives a negative reward which is calculated by the number of hotspots it contributes, times a scalar value of the cost per hotspot.

$$Rwd_i = \begin{cases} C & hotspots_i = 0 \\ -CpH * hotspots_i & hotspots_i > 0 \end{cases}$$

The reasoning here is similar to the one described in Section 6.2. In the first case, the function returns a positive feedback in order to reward the agent for not contributing to any hotspot. In the negative case the agent is penalized proportionally to the hotspots in which it contributes.

6.4 Solutions mixing level capping and ground delays

The approach here is identical to the one described in Section 6.2, but the initial state of the process for deciding delay regulations is the last state after agents have decided their - potentially revised- flight plans. Specifically, this is a two-stages process: In the first stage, after calculating level capping measures, agents' decide on their flight plans (as specified in Section 6.3). This new set of flight plans results in a relaxed DCB problem compared to the initial one, where some of the initial hotspots have been resolved. In the second stage, this DCB problem is solved by imposing ground delays to the decided flight plans, according to the method described in Section 6.2. Figure 6 illustrates this process.

This two-stages process results in a Pareto optimal solution regarding the two dimensions of decision making, w.r.t. delays and level capping measures. This is done without unnecessarily exploding the state-action space, as it would be done if we considered all the combinations of measures in the joint **State** × **Action** space. This can easily be seen if we consider that agents jointly decide the best possible level capping measures, resulting in a relaxed DCB problem that agents can resolve with joint decisions on ground delays. Less effective level capping measures would result in the need for additional ground delays for agents and less total reward, while in case the agents could decide on more effective level capping solutions, then they would do that as their reward would increase and they would not need to impose the minutes of delay decided.

7 Mimicking DRL models for the provision of explanations

The overall mimicking approach we follow in this work towards providing explanation content (interpretations) on individual agents' decisions at a fine level of granularity is shown in Fig. 7. Overall, given the well-trained DQN model, this is treated as an oracle for training a model realized by a

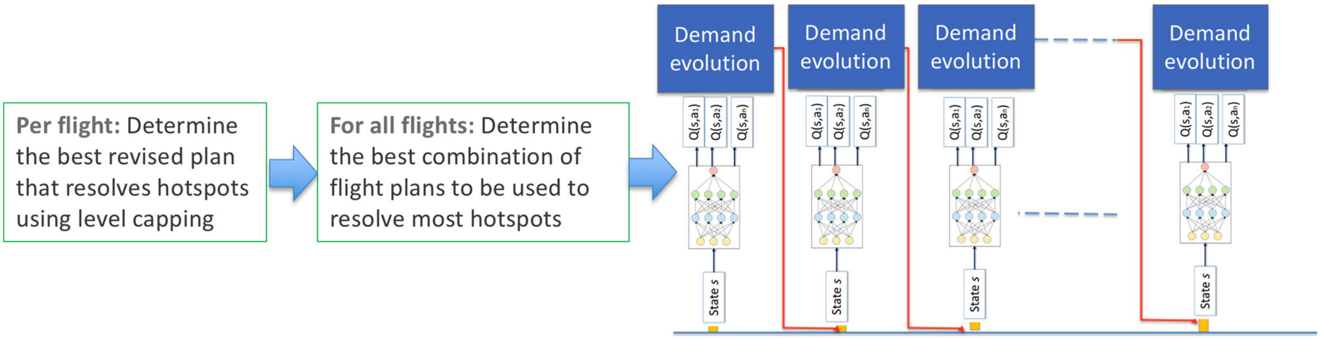

Per flight: Determine the best revised plan that resolves hotspots using level capping
For all flights: Determine the best combination of flight plans to be used to resolve most hotspots
Demand evolution
Demand evolution
Demand evolution
Demand evolution
Q(s,a₁)
Q(s,a₂)
Q(s,aₙ)
State s

Fig. 6 Combining level capping and ground delays measures

forest of Stochastic Gradient Trees (SGT), called the mimic model.

Specifically, as it is shown in Fig. 7, given an observation signal I, assuming in the general case that the full state may not be observable by the agent, the DQN model predicts the values of the actions a in that state, i.e. $Q(I, a)$. Given these values, together with the observation signal, the mimic model is trained to approximate the Q-value (denoted by $\hat{Q}$) for any possible option and state, following a regression process.

The mimic model predicted action is the one with the maximum $\hat{Q}$-value:

$$a^* = \arg\max_a \hat{Q}(s, a).$$

The fidelity of mimic model is determined by comparing the mimic model predicted action with the action prescribed by the DQN policy model, at each state and for any of the agents. We say that an action prediction is correct, if it coincides with the DQN prescribed action. We measure the accuracy of the mimic model as the number of correct predictions over all the predictions.

Subsequently, we provide background knowledge on SGT and specify how the mimic model is being trained and exploited to interpret agents' decisions per type of solution.

7.1 Stochastic gradient trees

Stochastic Gradient Trees (SGT) [64] is a state-of-the-art method for learning decision trees using stochastic gradient information as the source of supervision. SGT can operate in an incremental learning setting, although in our case we are using the method in a batch mode: As our interest is to mimic the trained DQN model, we aim to perform regression on the Q function, as it has been modeled by the online policy model of DQN. SGT have been shown to outperform state of the art regression trees.

Overall, we consider supervised learning settings where data at every timestep t is of the form $(x_t, y_t) \in X \times Y$, and the aim is to predict the value of y_t given x_t. In our case, as also discussed in the mimicking paradigm above, x_t is a (state, action) pair and y_t is the Q-value for that pair, as predicted by the DQN policy model. These samples are tagged with the simulation timestep and provided from each of the agents during simulation.

We succinctly describe how SGT incrementally construct decision trees, optimising arbitrary twice-differentiable loss functions. The main ideas behind SGTs are (a) evaluating splits and computing leaf node predictions using only gradient information, and (b) using standard one-sample t-tests to determine whether enough evidence has been observed to justify splitting a node.

Solving a regression problem in our case we aim to predict the $\hat{Q}$-value of each possible agent option, at any state: The objectives here is SGT (a) to learn how to rank alternative options per state, so as to (c) mimic faithfully (with fidelity) the decisions of the DQN model, and (b) to explain the reasons justifying each decision.

We train SGT using a squared error loss function

$$l(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2 \tag{12}$$

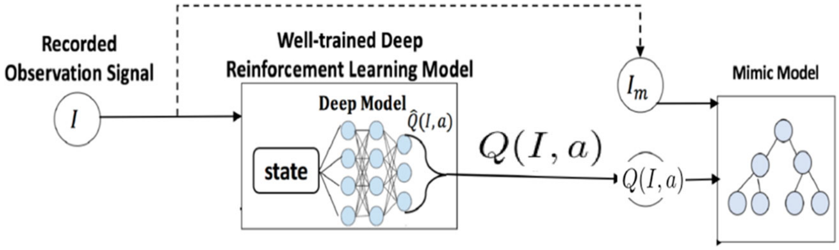

Recorded Observation Signal
I
Well-trained Deep Reinforcement Learning Model
Deep Model
state
$\hat{Q}(I,a)$
$Q(I,a)$
I_m
$\hat{Q}(I,a)$
Mimic Model

Fig. 7 The mimicking approach for interpretability

measuring how well our predictions, $\hat{Q}$, match the predictions Q provided by the deep model. Predictions are computed using SGT (function Q_{SGT}), optionally composed with an activation function, σ,

$$\hat{y} = \sigma(Q_{SGT}(x)) \tag{13}$$

The objective during training is to minimize the expected loss, as estimated from the data observed between the current timestep t, and the timestep r, at which the tree was previously updated. Assuming i.i.d. data, the expectation on the loss can be approximated in a stochastic manner using the most recent observations,

$$E(l(y, \hat{y})) = \frac{1}{t-r} \sum_{i=r+1}^{t} l(y_i, \hat{y}_i)^2 \tag{14}$$

Towards minimizing the expected loss, at each timestep, the method may modify the tree either by splitting one of its leaf nodes, or by updating the prediction made by a leaf node.

The empirical expectation of the loss function can be approximated using a Taylor expansion around the unmodified tree at time t (indicated by Q_{SGT}):

$$\mathcal{L}_t(u) \approx \sum_{i=r+1}^{t} [l(y_i, Q_{SGT}(x_i)) + g_i u(x_i) + \frac{1}{2} h_i u^2(x_i)], \tag{15}$$

where, u denotes the potential split given a new instance. g is the first derivative of the loss function w.r.t. $Q_{SGT}(x_i)$, and h the second derivative, i.e. $(\hat{y} - y)$ and 1 in our case, respectively.

Optimization can be expressed by eliminating the constant first term resulting in the following formula

$$\Delta\mathcal{L}_t(u) \approx \sum_{i=r+1}^{t} [g_i u(x_i) + \frac{1}{2} h_i u^2(x_i)] = \sum_{i=r+1}^{t} \Delta l_i(u) \tag{16}$$

that describes the change in loss due to a split u. The optimal splitting, given all possible splits is

$$v_u^*(j) = -\frac{(\sum_{i \in I_u^j} g_i)}{(l + \sum_{i \in I_u^j} h_i)} \tag{17}$$

where I_u^j is the set of indices of instances reaching the new leaf node identified by j, and v maps these new leaf node identifiers to the difference between their predictions and the prediction made by their parent. SGT use student's t-test to determine whether a split should be made, with the hypothesis that no split should be made. This avoids knowing in advance the range of values that can be taken for n terms $\Delta l_i(u)$.

The process to determine whether enough evidence has been collected to justify a split is prohibitively expensive to carry out every time a new instance arrives. Therefore, in practice, SGT check whether enough evidence exists to perform a split when the number of instances that have fallen into a leaf node is a multiple of some user specified parameter.

7.2 Training the mimic models

Following the mimicking approach with SGT as an interpretable model, we train an SGT mimic model to provide explanations on the measures decided by the deep model. For each of all the possible agent's options a, the regression process aims at minimizing the squared error loss function, as specified in formula (12). We use the Mean Absolute Error to measure the mean absolute difference between the Q-values provided by the DQN and the $\hat{Q}$-values predicted from the SGT.

For solutions with ground delays, described in Sections 6.2 and 6.4, each agent can decide among 11 different action options, corresponding to 0−10 minutes of additional delay at each timestep of the simulation. For any option and at any timestep, SGT provides an interpretation.

In the type of solutions where level capping measures are imposed as described in Section 6.3,

the agent can choose among two options corresponding to the original flight plan and the highly ranked plan with a level capping measure. In this case, SGT provides interpretations for agents' decision on imposing a level capping measure or not.

To train the mimic model the method gathers samples that correspond unique states visited over 400 episodes from all DQN agents interacting with the environment. To gather as many samples as possible (i.e., regarding different (state, action) pairs), the trained DQN interacts with the environment starting with exploration, having set the ϵ-greedy parameter $\epsilon = 0.9$ and diminishing ϵ as the episodes progress.

We train the mimic model passing over the training dataset once, i.e. in one epoch, taking advantage of the incremental learning abilities of SGT. To test the mimic models, we gather samples from all agents interacting with the environment in one episode with $\epsilon = 0.04$.

State features are discretized using equal width binning, with 64 bins. For the gradient updates, hyper-parameters are set as in [64]. To determine whether enough evidence has been collected to justify a split on a leaf node, the hyper-parameter regarding the number of instances that have fallen into that node is set to be a multiple of 200, as proposed in [64].

7.3 Interpreting agent's decisions

Interpretations comprise arguments regarding:

- Why the agent has decided an action;
- Why the agent should not take the decided action.

For the first, we exploit the stochastic gradient tree that corresponds to the demand measure decided by the DQN. The latter provides counterarguments for not taking the decided action. Counterarguments for an action are provided by an SGT model corresponding to any option, which is different from the one decided by the DQN.

For any specific state-action pair (s, a), the interpretation method follows the path in the stochastic gradient tree corresponding to action a from the root to the leaf node that makes the prediction of $\hat{Q}(s, a)$. The interpretation contains arguments for a decision. An argument corresponds to a stochastic gradient tree branch node and comprises a split feature and a split condition. The order of nodes visited, and thus the order of arguments in an interpretation, shows the importance of features to decisions.

As for example, let us consider the arguments made by SGT towards interpreting a decision for a flight to take no delay at a particular time instance: These are shown in order of importance as provided by SGT in Table 2.

We note that the same feature may appear multiple times in an explanation, as it can be associated with different conditions: This is the case for the "Existing delay" in our example. First, we have to note that the interpretation method provides the explanation content: This must be associated with the particular environment conditions, and the whole information is provided by visualizations and tools for rendering information and exploration of solutions.

Arguably, this is a detailed explanation for a single simulation step and from a single agent: Such explanations can be (a) associated with particular states of the airspace at specific time instances, to inspect the situation, (b) provided as a dataset for further exploration towards providing a qualitative understanding of agents' decision making. However, from this single explanation, one may understand that the model does not prescribe any additional minute of delay for that flight, given that it participates in 2 hostpots in the same sector (first two arguments), occurring early in the afternoon of that day (49th and 48th counting period, i.e. approx. at 16:30) while the flight has a small duration in specific sectors (5th and 8th arguments). To fully understand the situation one has to inspect the demand of the sectors during each period crossed by that flight: This information is provided by the visual analytics tools for exploring solutions described in the next section.

Table 2 Arguments for receiving no delay

Feature	Value
1st Hotspot	2nd sector (LECMBLI)
2st Hotspot	2nd sector (LECMBLI)
Counting period of 2nd Hotspot	49
Existing Delay (≤ 3.12)	0 min
Duration in 6th Sector	8 min (LECBP2R)
Counting period of 1st Hotspot	48
Existing Delay (≤ 1.56)	0 min
Duration in 2nd Sector	11 min (LECMASU)

A similar case, where the SGT argues for no minutes of additional delay for another flight is provided in Table 3. Actually, in this case, these are counterarguments provided by the mimic model, against the actual decision made by this flight for taking 1 min of additional delay. The arguments for that decision are presented in Table 4. If we consider the set difference of those arguments w.r.t. their order of importance, we can say that the decision of receiving an 1 minute of additional delay versus receiving no additional delay was based on (a) the number of hotspots the agent participates, (b) the counting period of its first hotspot and (c) the duration of the flight in the first sector crossed, which is LECBCCC. Contrarily, we can say that the first two hotspots that the agent participates and the counting period it stays in the second hotspot are considered as the most important factors for not taking any additional delay.

8 Visual analytics for exploring solutions and explanations

To promote trust in the use of the system and facilitate

- understanding of problematic situations requiring measures, i.e. demand-capacity imbalances;
- exploration and understanding of solutions developed by the multi-agent system;
- exploration and understanding of the explanations of the decisions taken by the individual agents,

we propose a Visual Analytics (VA) component, developed in accordance to general principles [65, 66] of visual data science and, more specifically, pattern search.

The VA component includes two modules providing different levels of abstraction and details:

Table 3 Arguments for not receiving any additional delay

Feature	Value
1st Hotspot	0nd sector (LECBCCC)
2nd Hotspot	1st sector (LECPL1W)
Counting period of 2st Hotspot	25
Counting period of 3rd Hotspot	26
Existing Delay	1min

Table 4 Arguments for receiving 1 minute of additional delay

Feature	Value
Number of Hotspots	3
Counting period of 1st Hotspot	26
Existing Delay	1min
Counting period of 3rd Hotspot	26
Duration in 1st Sector	19min (LECBCCC)

- **Solution Explorer**: high level of abstraction and low level of detail. It is meant for gaining overviews of the evolution of use of airspace over the scenario duration, identifying major differences between solutions, and tracking major changes along the process of solution development.
- **Sector Explorer**: lower level of abstraction and higher level of detail. Unlike Solution Explorer, Sector Explorer shows information about individual flights. This module also allows accessing all explanations for a selected flight, as provided by the mimic model.

To gain an overview of one solution, Solution Explorer proposes a table view demonstrated in Fig. 8. The columns of the table correspond to sectors and the rows to hourly counting periods. The cells contain horizontal bars with the lengths proportional to the counts of the flights that enter the sectors in the 1-hour periods corresponding to the rows. Where the number of the entries is over 110% of the sector capacity threshold, the cell contains a red vertical line crossing the bar showing the counts of the entries. The horizontal position of the line within the cell corresponds to the sector capacity; hence, it is possible to compare the capacity with the demand and judge the amount of excessive demand.

When the solution represented in the table view involves delayed flights, the bars inside the cells are divided into segments corresponding to different durations of the delays (Fig. 9). The segments are painted in different shades of grey, so that darker shades correspond to longer delays.

The VA component supports a variety of interactive operations, thus enabling focusing on particular solutions, selected time intervals and sectors of interest. Multiple solutions (e.g. initial state, an intermediate and final state) can be shown simultaneously for comparison, see Fig. 10.

Sector Explorer can present a single solution or provide a comparative view of two solutions (Fig. 11). A major part of the display is given to the main view where the horizontal dimension represents time. The time slider below the main view is used for choosing a time interval to be shown with the highest possible resolution. The vertical dimension of the main view accommodates several sectors one of which is considered as the focus sector. Above the band of the focus sector are the bands corresponding to the upstream sectors, i.e., the sectors from which flights come directly to the focus sector. The bands below the band of the focus sector correspond to the downstream sectors, i.e., the sectors to which flights go directly from the focus sector.

Apart from the flights, the main panel of Sector Explorer shows sector loads (i.e., demands) by time periods of 1-hour length, taken every step of 20 minutes. The demands are represented by time-based histograms with overlapping bars, as shown in Fig. 12.

The SectorExplorer component supports interactive selection of flights of interest. Data-driven selection is supported: for example, it is possible to select flights that were delayed or changed a sequence of crossed sectors from one solution to another. A list of selected flights and their properties can be presented visually in a tabular form Fig. 13. This table shows how delays for the selected flights were added over multiple steps of the model. For example, flight IBE3340 (on top of the table) got a total delay of 100 minutes in the result of 23 changes that occurred at steps 4 to 31.

As described in Section 7, each decision of the model is interpreted by a mimic model in the form of a list of conjunctive arguments. A collection of such arguments for a selected flight is present in a table, as shown in Fig. 14. Each row represents arguments for a decision made at some step. Column "Action" represents the added delay (from 0 to 10 minutes), while arguments like $featureA$ is in interval $[A_{min}..A_{max}]$ and actual feature values are displayed graphically in the columns of the table. Such tabular representation allows tracing the conditions over model's steps and enables an efficient assessment of consistency of decisions, assuming that similar actions are supposed to be performed under similar conditions.

To summarise, the VA component efficiently enables consideration of solutions from multiple perspectives: model evolution, sectors, hotspots, time intervals, and flights, and provides access to interpretations of model decisions. This information can be used by model developers for checking data quality and validating model outputs. Domain experts can use the proposed VA tools for gaining trust to models, understanding model behavior and for purposes of model certification. The VA component successfully implements major principles of human-centered machine learning [67]. A detailed overview of the proposed approach and its implementation is published in [68].

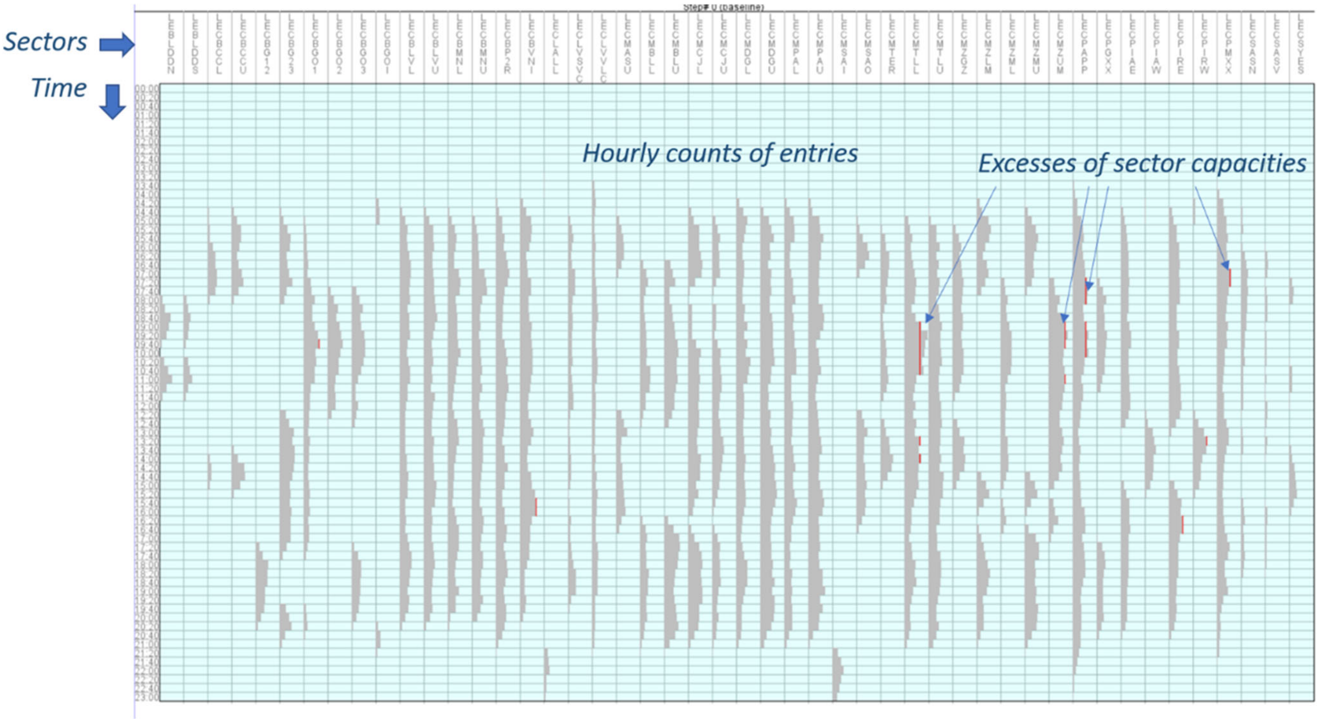


Fig. 8 An overview of a solution: a table view with columns corresponding to sectors and rows to time intervals

9 Experimental results

9.1 Data sources

For the construction of scenarios with a time horizon **H** of 1 day, both for training and testing the multiagent system, we employ a dataset reporting flight plans and sectorizations for the same spatial and temporal ranges.

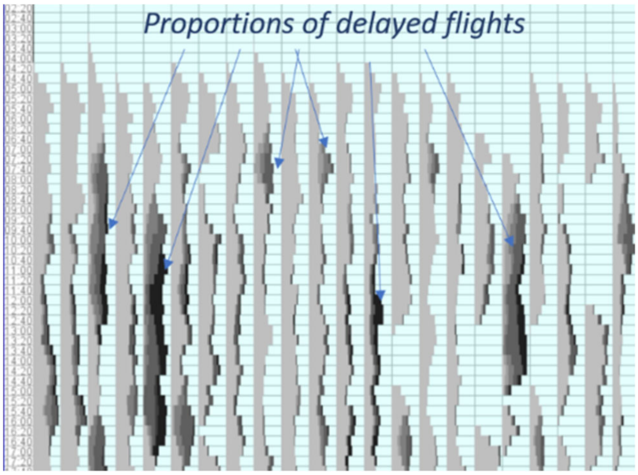


Fig. 9 An overview of a solution involving flight delays. Segmented bars of different shades of grey indicate proportions of delayed flights with different delay durations. Light grey corresponds to non-delayed flights; the darkness increases proportionally to the delay duration

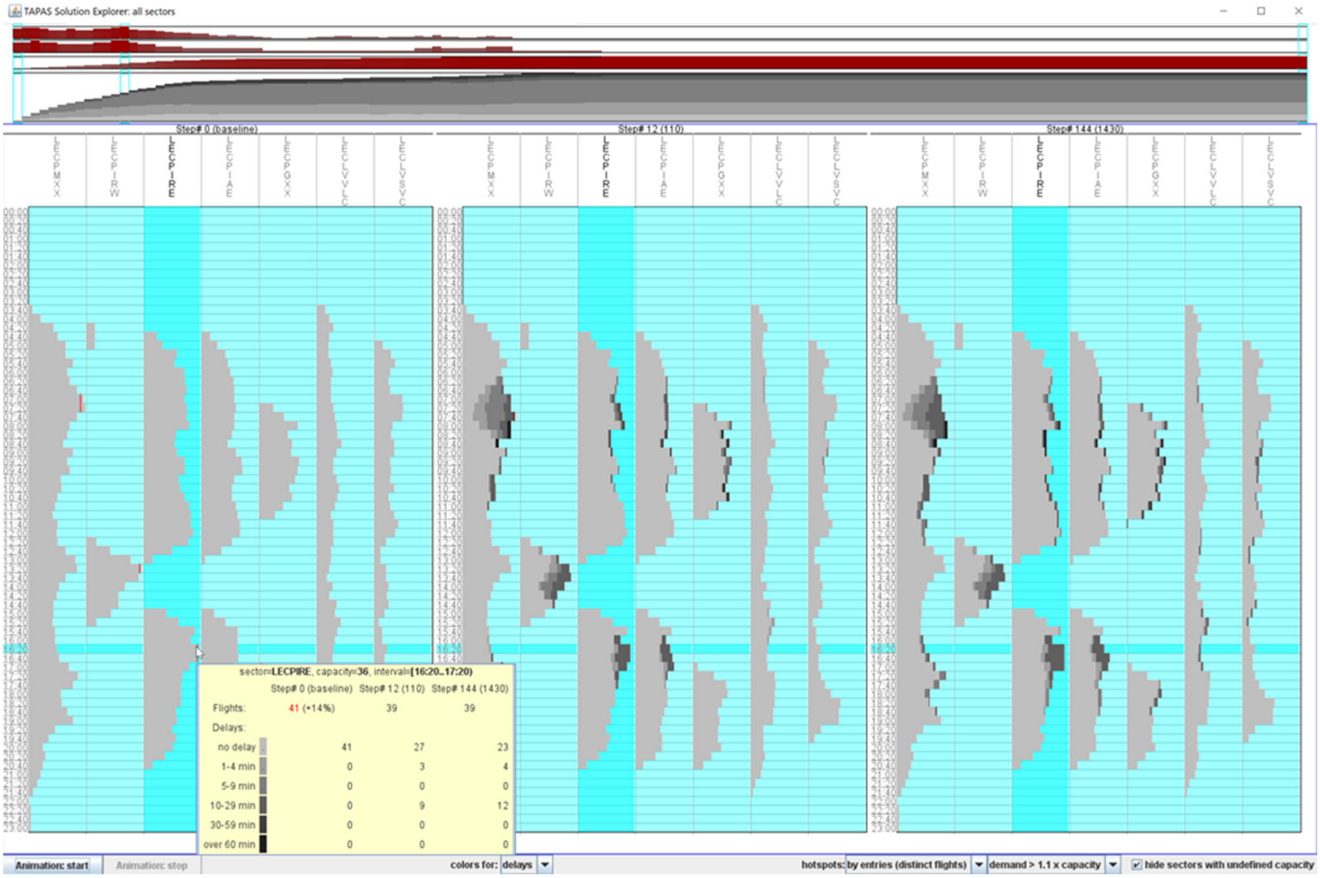
The flight plans dataset reports the intended trajectories of all the flights during 2019, worldwide. Data are provided in ALLFT+ format, described in [69]. For each flight we focus on the fields reporting the departure and destination airports, the aircraft callsign, the Last off-block time (LOBT) and the points profile. Each reported position in the point profile is defined by the latitude and longitude coordinates, as well as the intended flight level and the expected time of arrival to that position. The sequence of positions in the points profile, define the intended trajectory of the flight.

The sectorizations dataset provides the information about the activation intervals of airspace volumes. An Air Traffic Controller (ATC) is responsible for a specific airspace volume, which has an explicitly defined 3D geometry and an activation interval. A sector is considered active only during its activation interval.

The sectorizations dataset reports the active sectors in the European airspace, their capacities and their 3D geometries for the entire 2019.

9.2 Preprocessing of data

For the preparation of scenarios, we categorize the flight plans w.r.t. their departure and destination airports, and the geometry of the airspace of interest (the part that covers the Iberian Peninsula, namely, LTOT). Specifically, flights



sector=**LECPIRE**, capacity=**36**, interval=**[16:20..17:20)**

	Step# 0 (baseline)	Step# 12 (110)	Step# 144 (1430)
Flights:	41 (+14%)	39	39
Delays:			
no delay	41	27	23
1-4 min	0	3	4
5-9 min	0	0	0
10-29 min	0	9	12
30-59 min	0	0	0
over 60 min	0	0	0

colors for: delays

hotspots: by entries (distinct flights) | demand > 1.1 x capacity | ☑ hide sectors with undefined capacity

Fig. 10 Comparison of three solutions with a focus on a selected sector and the sectors connected to it

departing from airports within or close to the airspace of interest, can be potentially affected by delay and level capping measures. If only the destination airport is within LTOT (i.e. an incoming flight), or in case the flight crosses the airspace but neither the departure nor the destination airport is within or close to LTOT, only delay regulations can be applied. The rest of the flights are excluded from the dataset, since they are disjoint to the airspace of interest.

Regarding the sectorization data, we filter the sectors that overlap with LETOT and organize them in an STR-tree. Thus, given a spatio-temporal position (as part of the trajectory of a flight), we can identify the candidate sectors that cover the position by querying the STR-tree, and then refine the result by evaluating the activation intervals of the candidate sectors, and the 3D geometry of each sector. This process provides one active sector per trajectory point for original and revised flight plans.

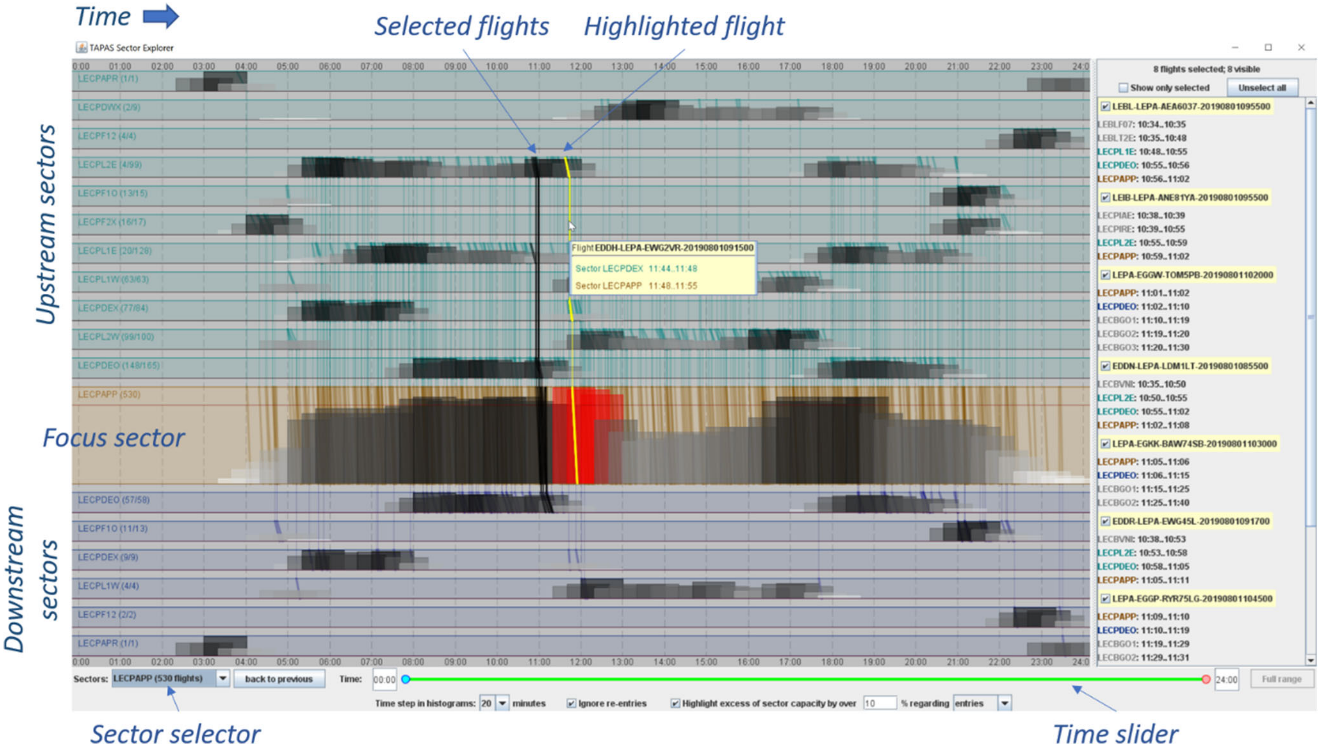
During the preprocessing phase, we compute all the revisions of an original flight plan that can be the result of applying any combination of time delay and level capping (where applicable) measures. We rank the flight plans with level caping measures w.r.t. the number and width of altitude changes. Formally, given a trajectory T crossing a set of sectors **R**, we compute the expected rate of flight-level change in all sectors (ERFL) by the formula:

$$ERFL(T) = \sum_{s}^{R} \frac{|\Delta alt_s^T|}{\Delta time_s^T} \tag{18}$$

where $|\Delta alt_s^T|$ is the difference of altitude at entry and exit points of T in sector s, and $\Delta time_s^T$ the time (in minutes) where T spends in sector s. We sum the absolute values of altitude changes, since any change (either upwards or downwards) should not cancel any previous changes. Finally, we rank the revised flight plans by their ERFL values, such that given two trajectories T_1 and T_2, T_1 is higher in the ranking iff $ERFL(T_1) < ERFL(T_2)$. The intuition behind this choice is to show higher preference to those trajectories that have less and smoother altitude changes.

9.3 Experimental settings and results of multiagent DQN

In our experimental settings, the DQN parameters are set as shown in Table 5. Specifically, regarding hyper-parameter β, it starts at 0.4 and is increased by 0.01 every 6 episodes,



Flight EDDH-LEPA-EWG2VR-20190801091500

Sector LECPDEX 11:44..11:48

Sector LECPAPP 11:48..11:55

8 flights selected; 8 visible

□ Show only selected Unselect all

☑ LEBL-LEPA-AEA6037-20190801095500
LEBLF07: 10:34..10:35
LEBLT2E: 10:35..10:48
LECPL1E: 10:48..10:55
LECPDEO: 10:55..10:56
LECPAPP: 10:56..11:02

☑ LEIB-LEPA-ANE81YA-20190801095500
LECPIAE: 10:38..10:39
LECPIRE: 10:39..10:55
LECPL2E: 10:55..10:59
LECPAPP: 10:59..11:02

☑ LEPA-EGGW-TOM5PB-20190801102000
LECPAPP: 11:01..11:02
LECPDEO: 11:02..11:10
LECBGO1: 11:10..11:19
LECBGO2: 11:19..11:20
LECBGO3: 11:20..11:30

☑ EDDN-LEPA-LDM1LT-20190801085500
LECBVNI: 10:35..10:50
LECPL2E: 10:50..10:55
LECPDEO: 10:55..11:02
LECPAPP: 11:02..11:08

☑ LEPA-EGKK-BAW74SB-20190801103000
LECPAPP: 11:05..11:06
LECPDEO: 11:06..11:15
LECBGO1: 11:15..11:25
LECBGO2: 11:25..11:40

☑ EDDR-LEPA-EWG45L-20190801091700
LECBVNI: 10:38..10:53
LECPL2E: 10:53..10:58
LECPDEO: 10:58..11:05
LECPAPP: 11:05..11:11

☑ LEPA-EGGP-RYR75LG-20190801104500
LECPAPP: 11:09..11:10
LECPDEO: 11:10..11:19
LECBGO1: 11:19..11:29
LECBGO2: 11:29..11:31

Sectors: LECPAPP (530 flights) back to previous Time: 00:00 24:00

Time step in histograms: 20 minutes ☑ Ignore re-entries ☑ Highlight excess of sector capacity by over 10 % regarding entries

Fig. 11 Components and contents of Sector Explorer in the mode of showing a single solution

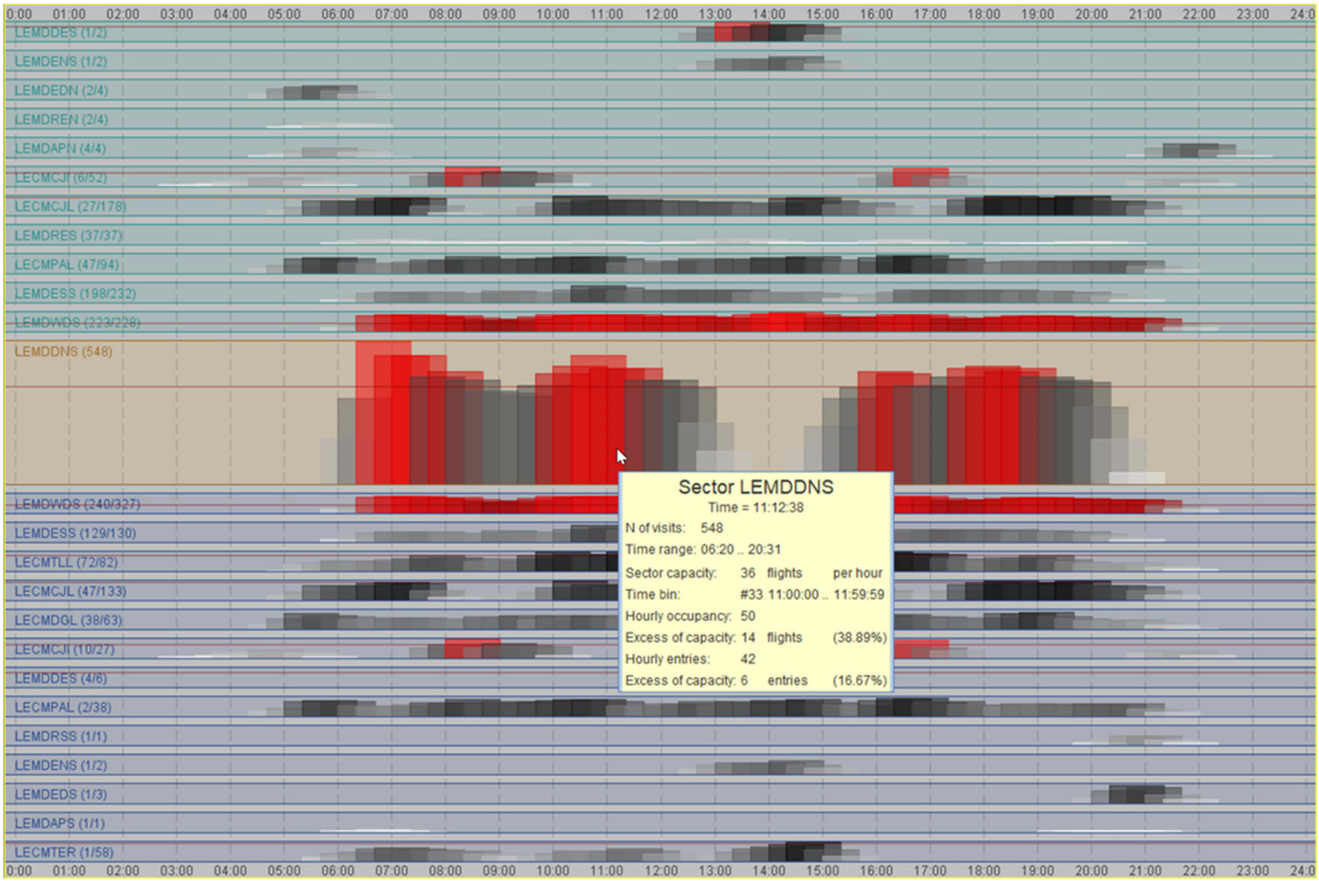
until it reaches the maximum of 1. Also, at the start of exploration, ϵ is set to 0.9 and is diminished by 0.01 every 15 episodes, until it reaches the minimum of 0.04. The total number of episodes is 2100. During testing, epsilon is set to 0.04.

All results presented are the averages from results produced by 10 independently trained models, each of which was tested 100 independent times.

Experiments were performed with six in total scenarios that correspond to days in 2019 with heavy traffic in the Spanich airspace: 20190622, 20190703, 20190704, 20190705, 20190708 and 20190714. These scenarios were selected based on the number of hotspots within the duration of the corresponding date. Specifically, the busiest Aeronautical Information Regulation and Control (AIRAC) days of 2019 for Madrid Area Control Center, from 20/06/2019 to 17/07/2019 have been identified. We selected the 6 out of these 28 days to test the proposed methods, based on the number of detected hotspots, while also selecting dates with different sectorizations in order to have a variety in the sectors to be monitored.

The testing environment and simulation have been introduced in Sections 6.2 and 6.3. The testing of the multi-agent system has been done in two ways, for all three types of solutions: First the policy model has been pre-trained in various scenarios. Those scenarios are 20190801, 20190802 and 20190803. The training was done sequentially, meaning 20190801 was used for training the first model, which initializes the training for 20190802 and the resulting policy produced initializes the training for 20190803. Then, the resulting policy model is evaluated in three individual demanding scenarios, showing the ability of the method to provide qualitative solutions by imposing delay regulations, level capping measures and combinations of these two types of measures. In a second group of experiments, the pre-trained policy model is further trained in a subset of scenarios in a gradual manner, and it is tested in two scenarios: In one included in the training set, and in a scenario not included in the training set. In so doing, we aim to show the ability and limitations of the proposed method to (a) learn policies that can be easily tuned to new scenarios, balancing between efficiency in training and quality of solutions, while agents accumulate knowledge as they are trained in different scenarios, and also (b) the possibility of method's performance deterioration, as experience is accumulated from scenario to scenario.

First, Table 6 presents data regarding each of three experimental scenarios: 20190705, 20190708 and 20190714. These are the most demanding scenarios and their results represent adequately the results from the other scenarios, which are used in the second group of experiments described subsequently. Flights column indicates the number of flights that cross LTOT during the specific day. Hotspots column indicates the number of hospots



Sector LEMDDNS
Time = 11:12:38
N of visits: 548
Time range: 06:20 .. 20:31
Sector capacity: 36 flights per hour
Time bin: #33 11:00:00 .. 11:59:59
Hourly occupancy: 50
Excess of capacity: 14 flights (38.89%)
Hourly entries: 42
Excess of capacity: 6 entries (16.67%)

Fig. 12 Hourly demands for sectors are represented by time-based histograms with overlapping bars

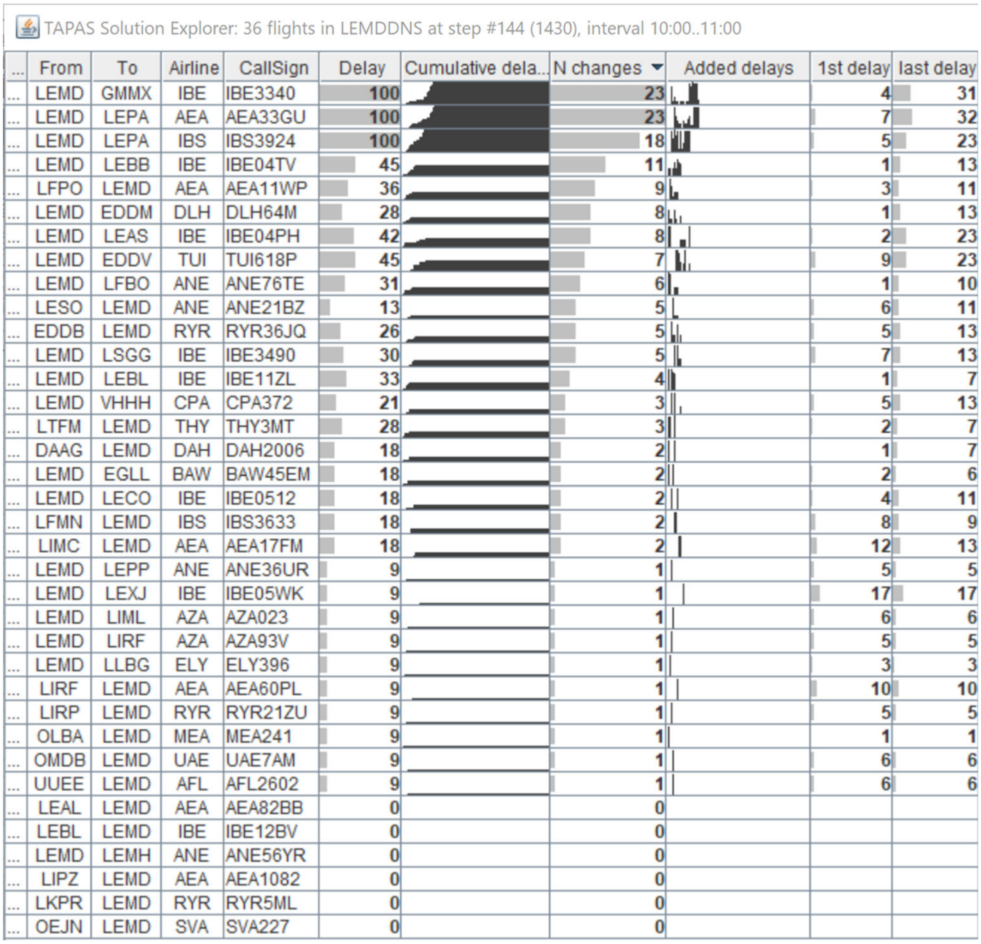
occurring in LTOT in the initial state of the scenario. Flights in hotspots column indicates the number of flights that participate in at least one hotspot. We can see that all three scenarios have an agent population of similar size, but 20190708 displays less severe traffic, with fewer hotspots and fewer flights participating in them. It must be noted here that all scenarios used are hand picked with traffic severity in mind, in order to pose the highest possible challenge to our approach.

Table 7 shows the first group of results: Solutions obtained by exclusively applying delay regulations. Final hotspots column shows the number of unresolved hotspots in the final state, after imposing the regulations. The month of July displays the most severe traffic of the year and this is evident especially in the case of 20190705 where, on average 38.4 hotspots remain unresolved. Average delay per flight column shows the total minutes of delay imposed, divided by the number of flights in the scenario. It must be noted here that all delays less than five minutes are ignored. Delayed Flights column shows the number of flights affected by more than four minutes of delay. The final column shows the ratio of total delay imposed to flights while ignoring delays less than five minutes, divided by the number of flights delayed (previous column). Figure 15 shows the box plots for all the previous indicators, per scenario. Horizontal lines of each box plot represent the 25th, the 50th, the 75th and the 100th percentile. Diamonds indicate outliers. It must be noted that the number of delayed flights is divided by 100, in order to maintain readability of the plot.

Table 8 shows solutions obtained by exclusively applying level capping measures. This type of measure is only applicable in a small number of flights, therefore, compared to delays, it has a smaller impact on the existing demand. The final hotspots column shows the number of unresolved hotspots in the final state. Affected flights column shows the number of flights with a level capping measure.

Table 9 shows results obtained by combining the two types of demand measures. Figure 16 shows the corresponding box plots, where the x axis specifies the indicator. The notation is identical to that of the previous plot.

Figures 15 and 16 provide a detailed view of the distributions of values regarding all indicators for all scenarios, facilitating the comparison between different types of solutions. Most notably, the combination of delay



TAPAS Solution Explorer: 36 flights in LEMDDNS at step #144 (1430), interval 10:00..11:00

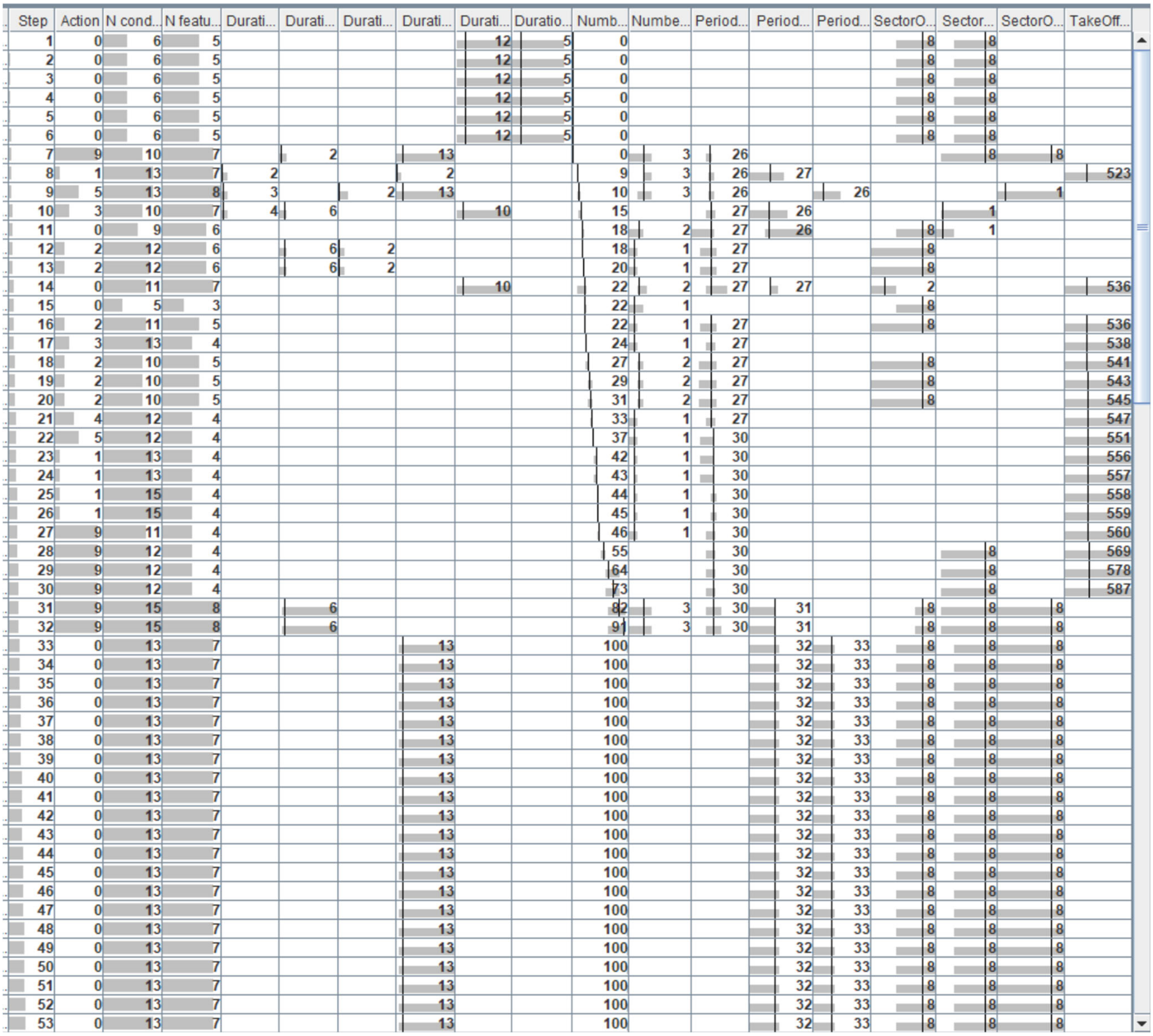
...	From	To	Airline	CallSign	Delay	Cumulative dela...	N changes	Added delays	1st delay	last delay
...	LEMD	GMMX	IBE	IBE3340	100		23		4	31
...	LEMD	LEPA	AEA	AEA33GU	100		23		7	32
...	LEMD	LEPA	IBS	IBS3924	100		18		5	23
...	LEMD	LEBB	IBE	IBE04TV	45		11		1	13
...	LFPO	LEMD	AEA	AEA11WP	36		9		3	11
...	LEMD	EDDM	DLH	DLH64M	28		8		1	13
...	LEMD	LEAS	IBE	IBE04PH	42		8		2	23
...	LEMD	EDDV	TUI	TUI618P	45		7		9	23
...	LEMD	LFBO	ANE	ANE76TE	31		6		1	10
...	LESO	LEMD	ANE	ANE21BZ	13		5		6	11
...	EDDB	LEMD	RYR	RYR36JQ	26		5		5	13
...	LEMD	LSGG	IBE	IBE3490	30		5		7	13
...	LEMD	LEBL	IBE	IBE11ZL	33		4		1	7
...	LEMD	VHHH	CPA	CPA372	21		3		5	13
...	LTFM	LEMD	THY	THY3MT	28		3		2	7
...	DAAG	LEMD	DAH	DAH2006	18		2		1	7
...	LEMD	EGLL	BAW	BAW45EM	18		2		2	6
...	LEMD	LECO	IBE	IBE0512	18		2		4	11
...	LFMN	LEMD	IBS	IBS3633	18		2		8	9
...	LIMC	LEMD	AEA	AEA17FM	18		2		12	13
...	LEMD	LEPP	ANE	ANE36UR	9		1		5	5
...	LEMD	LEXJ	IBE	IBE05WK	9		1		17	17
...	LEMD	LIML	AZA	AZA023	9		1		6	6
...	LEMD	LIRF	AZA	AZA93V	9		1		5	5
...	LEMD	LLBG	ELY	ELY396	9		1		3	3
...	LIRF	LEMD	AEA	AEA60PL	9		1		10	10
...	LIRP	LEMD	RYR	RYR21ZU	9		1		5	5
...	OLBA	LEMD	MEA	MEA241	9		1		1	1
...	OMDB	LEMD	UAE	UAE7AM	9		1		6	6
...	UUEE	LEMD	AFL	AFL2602	9		1		6	6
...	LEAL	LEMD	AEA	AEA82BB	0		0			
...	LEBL	LEMD	IBE	IBE12BV	0		0			
...	LEMD	LEMH	ANE	ANE56YR	0		0			
...	LIPZ	LEMD	AEA	AEA1082	0		0			
...	LKPR	LEMD	RYR	RYR5ML	0		0			
...	OEJN	LEMD	SVA	SVA227	0		0			

Fig. 13 List of flights with accumulated delays. The dynamics of delays accumulations is shown if columns "Cumulative delays" and "Added delays", representing total and momentary delays, accordingly

regulations and level capping measures not only provide better mean values for average delay and number of delayed flights, but also much less deviation from that mean, with remarkably less outliers. This reduction of measures comes at a slight cost in the number of unresolved hotspots in the case of 20190708 (4.5 additional hotspots to the 4.6 reported in the solution with delay regulations, out of the 82 in the initial state). In the other two scenarios the unresolved hotspots in the final state are practically the same for the two types of solutions. Minimal differences are also observed in average delay per delayed flight: for example we can see slightly better results in 20190714 in favor of the combined measures.

Therefore, experiments show that DQN can produce high-quality results, resolving the majority of the hotspots that occur in large airspaces under realistic conditions, applying combinations of measures. This is due to the ability of the method to "view" at each time step during simulation the demand for all sectors crossed by each agent, something that goes far beyond the cognitive abilities of humans to manage the inherent complexity at the extent that the system does with 7000 agents' joint decision. Such solutions, although not directly comparable to those provided by the Network Manager (i.e. those in Table 1), can extent our abilities to plan how to use the airspace well in advance, and in the best possible way, before applying further measures, e.g. assigning more ATCs.

In the next group of experiments we study the extent at which our approach can produce models that are transferable among scenarios, i.e. models which, when trained on a number of scenarios, can produce solutions in new scenarios with no or minimal training: This is essential to produce policies that can be easily tuned to new scenarios, balancing between efficiency in training and quality of solutions. To evaluate this possibility we compare the results presented in the previous Tables with those in Table 10. For all these experiments we utilized a model pre-trained with five scenarios: 20190622, 20190703, 20190704, 20190705 and 20190708. This model is denoted M20190622-0708, as it is constructed gradually to scenarios with respect to their chronological order: it is constructed by training a first model in the first scenario, the resulting model is further trained in the second scenario, and so on. The individual models per scenario are indicated by M2019X. All of the scenarios are hand-picked and present exceptionally severe traffic in LTOT. The main goal here is to test



Step	Action	N cond...	N featu...	Durati...	Durati...	Durati...	Durati...	Durati...	Duratio...	Numb...	Numbe...	Period...	Period...	Period...	SectorO...	Sector...	SectorO...	TakeOff...
1	0	6	5					12	5	0					8	8		
2	0	6	5					12	5	0					8	8		
3	0	6	5					12	5	0					8	8		
4	0	6	5					12	5	0					8	8		
5	0	6	5					12	5	0					8	8		
6	0	6	5					12	5	0					8	8		
7	9	10	7		2		13			0	3	26				8	8	
8	1	13	7	2			2			9	3	26	27					523
9	5	13	8	3		2	13			10	3	26		26			1	
10	3	10	7	4	6			10		15		27	26			1		
11	0	9	6							18	2	27	26		8	1		
12	2	12	6		6	2				18	1	27			8			
13	2	12	6		6	2				20	1	27			8			
14	0	11	7					10		22	2	27	27		2			536
15	0	5	3							22	1				8			
16	2	11	5							22	1	27			8			536
17	3	13	4							24	1	27						538
18	2	10	5							27	2	27			8			541
19	2	10	5							29	2	27			8			543
20	2	10	5							31	2	27			8			545
21	4	12	4							33	1	27						547
22	5	12	4							37	1	30						551
23	1	13	4							42	1	30						556
24	1	13	4							43	1	30						557
25	1	15	4							44	1	30						558
26	1	15	4							45	1	30						559
27	9	11	4							46	1	30						560
28	9	12	4							55		30				8		569
29	9	12	4							64		30				8		578
30	9	12	4							73		30				8		587
31	9	15	8		6					82	3	30	31		8	8	8	
32	9	15	8		6					91	3	30	31		8	8	8	
33	0	13	7				13			100			32	33	8	8	8	
34	0	13	7				13			100			32	33	8	8	8	
35	0	13	7				13			100			32	33	8	8	8	
36	0	13	7				13			100			32	33	8	8	8	
37	0	13	7				13			100			32	33	8	8	8	
38	0	13	7				13			100			32	33	8	8	8	
39	0	13	7				13			100			32	33	8	8	8	
40	0	13	7				13			100			32	33	8	8	8	
41	0	13	7				13			100			32	33	8	8	8	
42	0	13	7				13			100			32	33	8	8	8	
43	0	13	7				13			100			32	33	8	8	8	
44	0	13	7				13			100			32	33	8	8	8	
45	0	13	7				13			100			32	33	8	8	8	
46	0	13	7				13			100			32	33	8	8	8	
47	0	13	7				13			100			32	33	8	8	8	
48	0	13	7				13			100			32	33	8	8	8	
49	0	13	7				13			100			32	33	8	8	8	
50	0	13	7				13			100			32	33	8	8	8	
51	0	13	7				13			100			32	33	8	8	8	
52	0	13	7				13			100			32	33	8	8	8	
53	0	13	7				13			100			32	33	8	8	8	

Fig. 14 Decisions applied to a single flight over multiple model steps are explained by a series of sets of arguments. Arguments, intervals of feature values, and actual values for each step are shown in table rows

the M20190622-0708 model against scenario 20190714, which was not used in the training stage. In so doing, we can explore the value of accumulated knowledge from training with different scenarios. In addition to that, we test M20190622-0708 to scenario 20190705, which is a scenario used in the training stage, but not the final one. Thus, we can evaluate the possible deterioration in performance, as the model is trained in additional scenarios and accumulates new experience on top of that gained in a past scenario.

We use three different "modes" in these experiments: (a) No additional training of M20190622-0708 to the specific scenario in which it is tested, noted with "no training" in the Training Episodes column in Table 10. (b) Minimal-training of M20190622-0708 with 500 exploitation episodes, denoted with 500. (c) Half-training of M20190622-0708 with 1000 exploitation episodes, denoted with 1000. Minimal and half-training are with respect to the number of episodes of the original experiments, which where 2100. The rest of the columns report on the indicators reported also in the previous experiments.

As shown in Table 10, there is merit in the pre-training scheme. In scenario 20190714, the model produces a solution with 23.7 unresolved hotspots, out of the 92 in the initial state, with no additional training: A solution that leaves 20.5% hotspots occurring in the initial state unresolved, in addition to those not resolved

Table 5 Hyper-parameters

Hyper-parameter	Delay solutions	Level Capping solutions
fully connected layers	4	2
activation	relu	relu
nodes	512	100
loss	Huber	Huber
Huber δ	0.05	0.05
batch size	200000	32
optimizer	Adam	Adam
α	0.0001	0.0005
epochs	20	10
clipnorm	0.1	0.1
γ	0.99	0.99
τ	0.9	0.9
replay memory size	3000000	3000
memory's ϵ	0.05	0.05
memory's α	0.6	0.6
memory's β min	0.4	0.4
memory's β max	1	1
starting ϵ	0.9	0.9
min ϵ	0.04	0
training episodes	2100	2000

by the M20190714 model. This shortcoming can be mitigated with minimal additional training. With minimal training, the M20190622-0708 model reaches very similar effectiveness compared to that of M20190714. Specifically, the M20190622-0708 model leaves 0.2% of the hotspots unresolved, in addition to those not resolved by the M20190714 model, while imposing 25% more delays. Half training improves these results even further. As shown in Fig. 17 the M20190622-0708 model produces results that are comparable to the results of M20190714, with less variation.

Regarding the scenario 20190705, and in comparison to the results reported with the model M20190705, the model M20190622-0708, with no additional training, produces a solution that leaves 4% hotspots occurring in the initial state unresolved, in addition to those not resolved by the M20190705 model, while imposing 7% more regulations. Thus, it seems that there is some deterioration in performance in case a model trained in a scenario is trained with additional scenarios. This is something that needs further investigation. Additional training seems to close the gap between the two models further. Half-training results in a penalty of 1% unresolved hotspots in addition to those not resolved by the M20190705 model, and 4% more regulations.

Table 6 Scenarios' data

Scenario	Flights	Initial Hotspots	Flights in Hotspots
20190705	6676	100	2074
20190708	6581	79	1567
20190714	6773	92	2004

As can be observed in Fig. 18, the values for all the indicators reported by the M20190622-0708 model are very close to the results reported by the M20190705 model, except for the unresolved hotspots in the case of no training: Indeed, the M20190622-0708 model produces results with less variation compared to those of M20190705, while M20190622-0708 values are gathered in the upper half of the M20190705 plots, and in the lower half for the average delay per delayed flight.

Figures 19 and 20 delve further in comparing the effectiveness of different approaches in scenario 20190714. Figure 19 shows the distribution of delay minutes to flights. These results are provided for the solutions with delay regulations and solutions that combine delay regulations and level capping measures, solutions produced by the M20190622-0708 model with no follow-up training (no training) and solutions produced with half training (i.e. 1000). Figure 20 shows the distribution of unresolved hotspots to flights, for the different types of solutions produced by the corresponding models. First, as shown in Fig. 19, M20190622-0708 with half training manages to reduce the distribution of delays to flights compared to the "no training case". However it increases considerably the number of flights in the range 10-59min and slightly in the other delay ranges, compared to the solutions with delays (D) and mixture of measures (D+LC). Also, as shown in Fig. 20, the M20190622-0708 model with half training is much more effective than with no training in resolving hotspots.

As a conclusion for this group of experiments, DQN manages to accumulate knowledge gradually while the model is being trained in different scenarios, however additional training is needed to "tune" the policy to a scenario. The additional training time required is considerably less than the time needed for training a model from scratch to any specific scenario. Thus the pretraining scheme produces models that can balance between the training time and the quality of the provided solutions. In addition to that, successful accumulation of new knowledge while training models in subsequent scenarios needs further investigation.

9.4 Evaluation of the mimic model

Given the models learned by training DQN in each of the individual scenarios, i.e., M20190622, M20190703, M20190704, M20190705, M20190708, M20190717, we train several mimic models as follows: (a) One model for

Table 7 Solutions with delay regulations

Scenario	Final Hotspots	Average delay per flight	Delayed flights	Average delay per delayed flight
20190705	38.4	13.04	1556.5	56.15
20190708	4.6	11.4	1387.2	54.21
20190714	4.8	10.72	1645.2	44.03

each of these individual scenarios, denoted by MM2019X, and (b) one model per subset of models, with respect to the chronological order of scenarios, denoted by MM2019X-Y. For instance, MM20190622-0708 denotes the mimic model trained using samples from all DQN models M20190622 to M20190708, in order. All these models have been trained to regulate flights with ground delays only, while results are provided by considering predictions per delay option in {0, 1, 2, 3...10} for solutions with delay regulations and combination of delay with level capping measures. In so doing, we aim to show the fidelity of the mimic models to the DQN models, as well as the ability of SGT to accumulate new knowledge and generalize beyond specific training scenarios.

First, for each mimic model we report the max tree depth in Table 11. Although not an indicator that can clearly show the complexity of the explanations, the tree length provides a measure indicating the amount of information one can expect from any explanation. Table 11 further shows how adapting a trained MM2019X model to subsequent models makes the explanations more lengthy, due to the need of SGT to adapt the accumulated knowledge to the new data. Therefore, as Table 11 shows, trees' depth for the MM2019X individual models (1st column) is maximum 28 and in average approximately 26. Training these models further in subsequent scenarios, the depth of trees increases consistently, and can increase considerably, up to 35 in the worst case, and in average up to approx. 33 (last column). It must be noted that the model increases considerably its length when training MM20190622-0704 with samples from M20190705, i.e. in MM20190622-0705, which is not the case between other model "transitions".

To measure the fidelity of mimic models, we measure the accuracy of the predictions made w.r.t. the predictions made by DQN. Specifically, we report on the accuracy per delay option, in solutions with delay regulations (D), and in solutions with combinations of delay regulations and level capping measures (D+LC). The results are presented in Table 12 and show the remarkable fidelity of the mimic models to the DQN models.

Then, we gradually train a mimic model in the first four scenarios, given the corresponding DQN models for imposing delay regulations, for each of these scenarios. Next we measure the fidelity of each mimic model to the corresponding DQN models for ground delays (D) and combination of delays with level capping measures (D+LC). This is essential to produce interpretable models

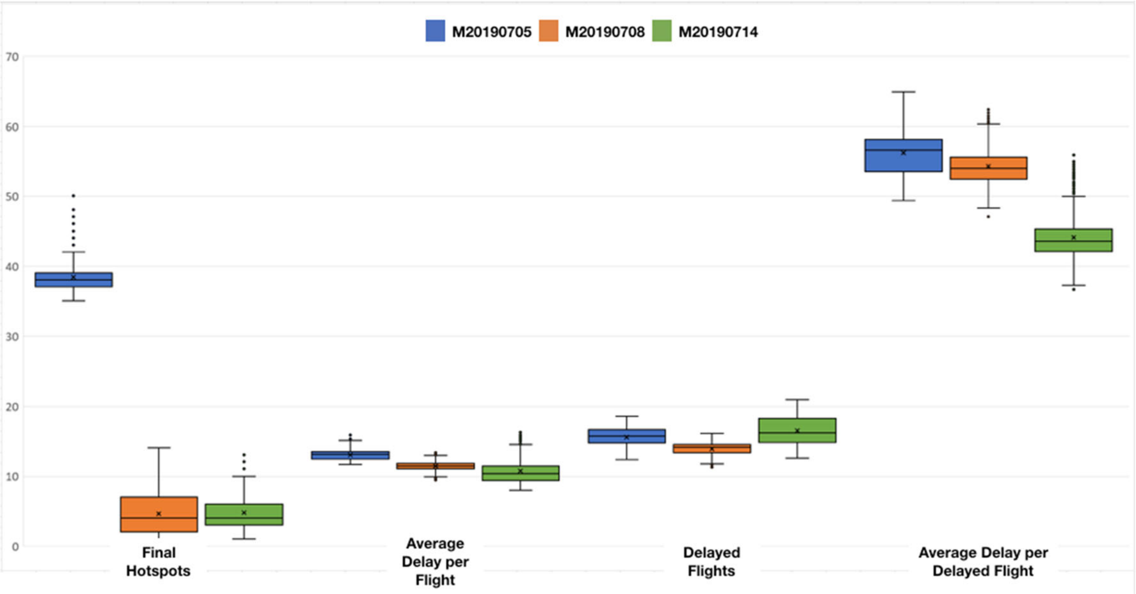


Fig. 15 Box plots for delay regulations

Table 8 Solutions with level capping measures

Scenario	Final hotspots	Affected flights
20190705	90	59.3
20190708	76	2.6
20190714	77	113.2

that can accumulate successfully knowledge from DQN policy models, with high fidelity.

Results in Table 13 show that SGT manage, due to their incremental learning abilities, to accumulate new knowledge and provide very accurate results, also in comparison to the fidelity of models trained per scenario (Table 12).

Next, given the mimic model MM20190622-0705, we measure the fidelity of that mimic model to the DQN models for delay regulations (D) and combinations of delays with level capping measures (D+LC) for each of the scenarios. In so doing, we aim to show whether accumulating new knowledge leads to reducing the accuracy of the mimic model to previously seen scenarios, and whether the mimic model can demonstrate satisfactory fidelity to unseen scenarios. The results are presented in Table 14: These show that indeed, mimic models accumulate successfully new knowledge, as they consistently improve the accuracy to seen scenarios while being trained to new ones, and they do have the ability to generalize so as to provide highly accurate results to unseen scenarios, i.e. to 20190708 and 20190717, in comparison to the accuracy of MM20190708 and MM20190717, respectively, shown in Table 12.

To further delve in the predictions made by the mimic models, Table 15 reports on the mean absolute error, per model,

considering the predicted $\hat{Q}$ values from the mimic model and the corresponding Q values provided by the DQN model per delay option.

As results on absolute errors show, while the mimic model is being trained accumulating new knowledge from subsequent DQN models, the mean absolute error (MAE) is being increased, with the exception of training MM20190622-0703 with samples from M20190704: In this case (MM20190622-0704) MAE is reduced, which shows that the increase of mimic model size reported earlier resulted to a good fit in the new samples.

9.5 Validation exercises with human operators

Validation experiments aim to qualitatively evaluate how effective the “explanatory” capabilities of the automation tools (i.e. DRL agents, with the mimic model) would be in operational situations. The experiments were carried out using a full, Human-in-the-Loop, interactive simulation environment that allowed the automation tools to be integrated in operational exercises where DCB measures were required. To ensure that the situation was as realistic as possible, the INNOVE Network Management simulation platform[2] was used to create validation scenarios for six scenarios taken from traffic days in the July 2019 time period. These days were selected since the summer period is the most heavily loaded in the Spanish ATM system, and as a result, the region experiences many hotspots. To respond to those hotspots, the automation tools propose solutions, that spread the demand from the overloaded sector(s) and period(s) to other sectors and later periods that are less busy, as described in Sections 6.2, 6.3, 6.4. Using a specially adapted Flow Manager Position tool (the FMP client), the DCB measures being proposed by the automation tools are published in the emulation of the environment that the Network Manager uses today, being simulated by INNOVE – as a set of measures to flights (using delays that have been assigned based on agents’ decisions). Human operators then receive information on the problem and solutions in parallel from the INNOVE (via the FMP client) and the Visual Analytics (VA) tools.

Depending on the level of automation selected during the validation exercise, the DCB process was either carried out in collaboration with the human operators or fully automatically, as follows:

In *collaborative mode*, the operator was able to review the proposed solutions in a ‘what-if’ sandbox environment before selecting part or all of the solution set and publishing the proposed actions to implement the proposed measures(s). Once published, the results of the selected actions could be immediately seen in the FMP client traffic demand charts (such as the one shown in Fig. 1).

In full *automation mode*, the solutions identified by the automation tools were automatically converted into the corresponding Network Manager requests and were published to INNOVE without human intervention. Thereafter the operators were required to consult the FMP client traffic demand charts and additional explanatory information provided via a co-located Visual Analytics (VA) tools in order to develop an understanding of what had happened, and why.

The purpose of the validation exercises was not specifically intended to evaluate how the automation components performed, although as already described, a separate evaluation was carried out to see how the automation tools performed in the chosen scenarios, and under different training conditions. Instead, the main objective was to try to evaluate how the information being

[2] Information about INNOVE is available at https://www.eurocontrol.int/simulator/innove and https://www.isa-software.com/innove-23-deployed-at-eurocontrol/

Table 9 Solutions with delay regulations and level capping measures

Scenario	Final hotspots	Average delay per flight	Delayed flights	Average delay per delayed flight
20190705	38.2	11.80	1357.5	58.17
20190708	8.9	11.05	1317.1	55.28
20190714	5.4	8.08	1360.2	40.48

gathered and published by the explanatory component (i.e. the mimic model) could be best used through the advanced VA tools to promote a high level of understanding and situational awareness for the operators when the "black-box" automation components were solving problems in place of the human. In order to evaluate how effective the explanatory components had been, a combination of "over-the-shoulder" observation, post exercise de-briefs and questionnaires were used to help measure levels of understanding, emphasizing on confidence in the system and situational awareness for the human operators. Considering the number of operators involved in the validation experiments (which was held over three days in June at the research premises of the Spanish Air Navigation Service Provider – CRIDA), we aimed at a qualitative validation process. We must emphasize the high-level of expertise of operators involved in the exercises and thus, their informed feedback. The exercises provide some interesting feedback, in particular relating to the levels of explanation offered by the mimic model with the support of VA tools.

In order to evaluate the effectiveness of the explanatory components, a combination of "over-the-shoulder" observation, post exercise de-briefs and questionnaires were used to help measure levels of understanding, levels of confidence in the system and situational awareness for the human operators.

Considering the number of operators involved in the validation experiments (which was held over three days in June at the research premises of the Spanish Air Navigation Service Provider – CRIDA), the analysis was limited to a qualitative validation process. Nevertheless, the expertise of the participants in the domain was very high so all feedback was of high value.

The exercises provide some interesting feedback, in particular relating to the levels of explanation offered by the mimic model with the support of VA tools.

Operators were given two days to become acquainted with tools, performing training exercises with manual operation for resolving hotspots or with the help of automation tools to identify hotspots, to resolve DCB problems and explore solutions with the VA tools. In total 8 exercises were finally performed in manual mode (i.e. as it is done today), in collaborative and in automation modes, allowing deep dive into the advanced features provided by the VA tools.

It must be noted that, as the validation was being performed with humans, but at a very low technological

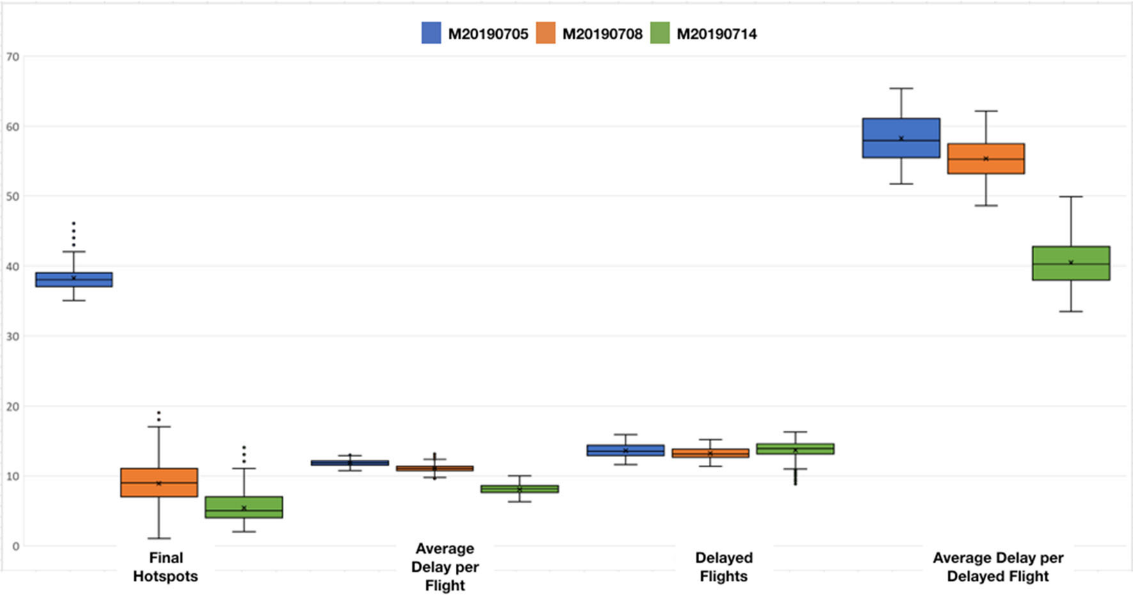


Fig. 16 Box plots for delay regulations and level capping measures

Table 10 Results from the M20190622-0708 model

Scenario	Training Episodes	Final Hotspots	Average delay per flight	Delayed Flights	Average delay per delayed flight
20190705	no training	42.6	13.94	1798.7	51.75
20190705	500	40.3	13.97	1792.5	52.03
20190705	1000	39.4	13.60	1752.7	51.81
20190714	no training	23.7	12.43	1997.5	42.17
20190714	500	5.0	13.40	2078.1	43.63
20190714	1000	4.3	13.03	2082.2	42.42

readiness level, it was not required to perform tests or gather metrics that would be essential at a later stage in the development of the automation and VA tools (e.g. certification and safety testing prior to the operational deployment).

The main findings regarding these exercises are summarized as follows:

- Users indicated that while it was very interesting to use the VA tools to explore the highly detailed information regarding solutions and explanations during the familiarisation (humans' training) processes, once scenarios were being executed in an operational context, the user tended to consult these features considerably less than during the training.
- However, despite this general observation, it was noted that when solutions developed by the automation tools were highly complex, and based on solutions to multiple constraints (rather than 'one-by-one' which is the typical approach used by operators on a daily basis) and for a much larger region of interest (the automation tools solved DCB problems for all of Spain, whereas the human operator would focus only on their own area – e.g. Madrid Area Control Centre), then the operator did tend to consult the more detailed VA explanatory features to try to better understand why a given situation had occurred (e.g. a flight may have received a measure because, although it did not participate in any hotspot in the initial problem state, because it was involved in hotspots that emerged due to measures to other flights).
- Detailed VA features were very useful during familiarisation with the tools, but users pointed out that these were less likely to be consulted in operational use: Building trust on the automated system is crucial for its operational use, and this mainly depends on the quality of the solutions provided (which however has to be built via the appropriate explainability facilities).

Regarding the modes of exercises, the main remarks are as follows:

- *Manual* mode: It is very difficult to manually solve all issues, even if one considers only the Madrid airspace, since, too many manual processes were

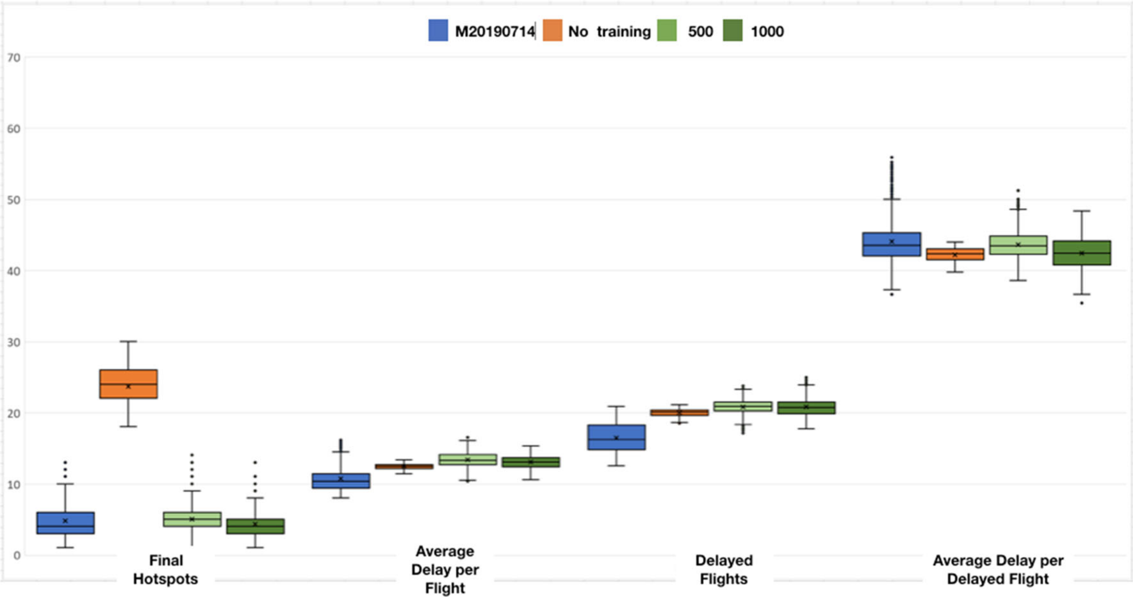


Fig. 17 Box plots for M20190622-0708 results in scenario 20190714

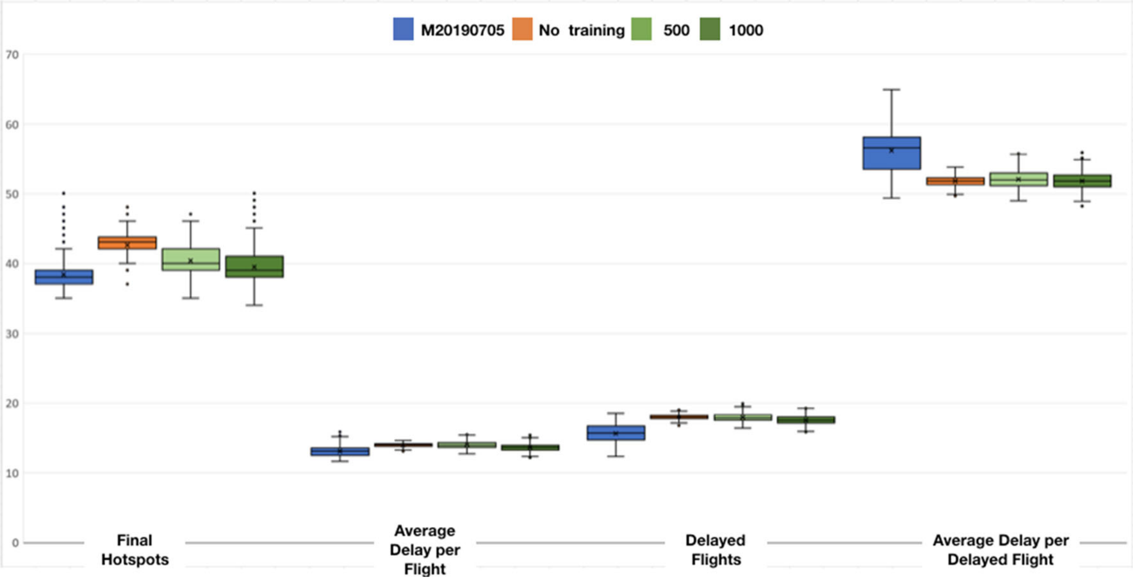


Fig. 18 Box plots for M20190622-0708 results in scenario 20190705

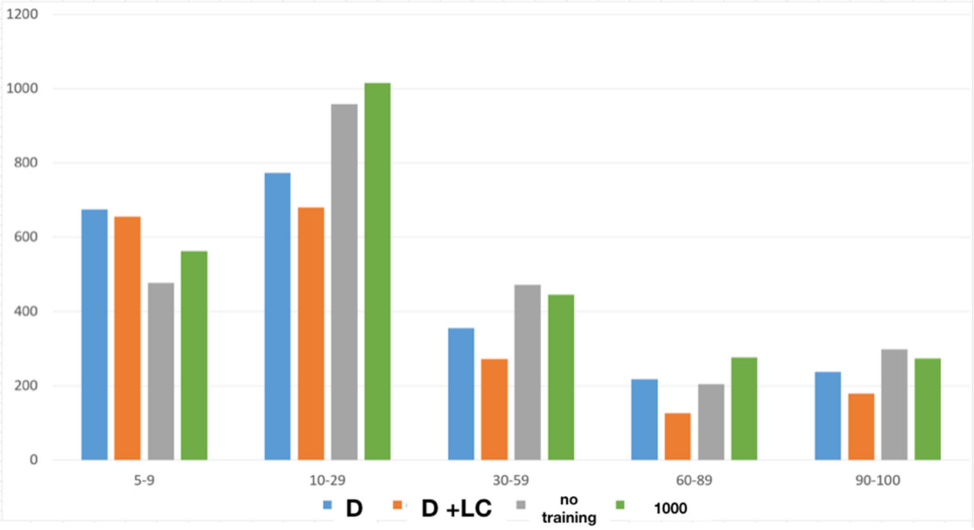
required. Problems would have been solved by the tools that the NM uses today and more complex problems were difficult to solve 100% manually. Smaller/simpler overloads were able to be solved by the human operator in an efficient manner.

- *Collaborative* mode: Operators were able to solve more complex hotspots using proposed solutions. However if only a subset of proposed solutions was selected, then solutions were not always successful: This proves the necessity of the measures in solutions, but also the complexity of solutions and the need of solutions traceability, as part of the explanations provided.
- *Automation* mode: Solutions proposed for most of the problems and for low/medium overloads worked efficiently. However, for high overload or overload for a sustained period, solutions created very high delay. In practice today heavy overloads are solved by additional measures (e.g. airspace configuration changes).

Furthermore, operators indicated that:

- A higher level of interaction with the automation tools is required, so as to be able to select a subset of measures for some of the flights, review the situation and proceed with additional solutions for those issues that remain.
- More aggregated interpretations of agents' decisions would be beneficial and would help them to increase their confidence to the solutions proposed, mastering

Fig. 19 Histogram of M20190622-0708 distribution of delays in 20190714



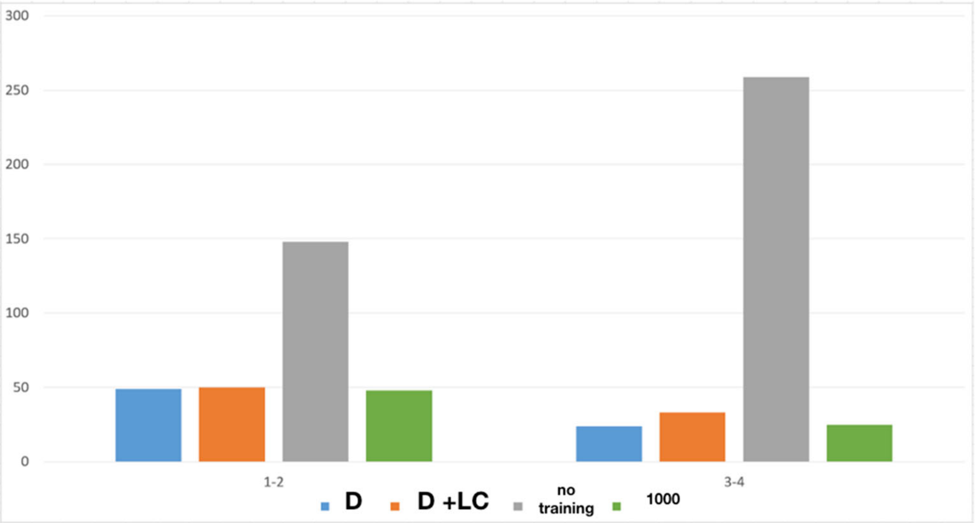


Fig. 20 Histogram for M20190622-0708 distribution of unresolved hotspots in 20190714

the inherent complexity, as solutions may be due to complex phenomena that are hard to be traced.

Overall, automation tools brought a shift to the operators' paradigm for resolving DCB problems. Solutions proposed by the automation tools were disruptive (in terms of the solutions operators are used to, today) due to the spatial and temporal extent in which the automation tools operate, and due to the inherent complexity of the DCB process in that scale of operations. Explanations at a fine granularity and scale provide useful information that operators can use offline (e.g. in the training process), given ample time. In the operational process, interpretations should be aggregated at a more appropriate level (this level needs to be investigated and specified), facilitating comprehensibility, mastering of complexity and tracing of solutions.

Table 11 Depth of SGT model

	Models			
Delay option	MM20190622	MM20190622 -0703	MM20190622 -0704	MM20190622 -0705
0	25	28	35	35
1	27	29	31	33
2	24	26	30	32
3	25	28	32	34
4	26	29	32	34
5	25	27	31	33
6	25	27	33	34
7	24	26	31	32
8	28	31	33	33
9	26	30	35	35
10	26	27	29	31

10 Conclusions

In this work we address the challenging issues of scalability and complexity towards advancing automation in real-world multiagent settings with thousands of agents, aiming to (a) compute qualitative solutions to congestion problems that arise naturally in the Air Traffic Management domain whenever demand of airspace use exceeds capacity, resulting to *"hotspots"*; and (b) provide explanations on how individual agents' decisions jointly affect their common setting.

Specifically, this paper (a) demonstrates how DQN can be used to address scalability in complex large-scale multi-agent settings, providing a policy model for agents to jointly decide on different types of hotspot resolution measures; (b) proposes the use of Stochastic Gradient Trees to build interpretable models that mimic the decisions of the well-trained DQN model, providing explanation content for agents' in fine granularity and scale; and (c) proposes visualization methods and visual analytic tools for rendering and exploring explanations content and solutions, addressing the scale of the multi-agent task, and the complexity of the problem.

Table 12 Accuracy of predictions achieved by models trained on individual days

Day	Mimic Model	D	D+LC
20190622	MM20190622	96.57	94.48
20190703	MM20190703	94.11	97.56
20190704	MM20190704	90.01	92.70
20190705	MM20190705	92.76	95.96
20190708	MM20190708	94.65	92.38
20190717	MM20190717	89.54	94.21

Table 13 Accuracy of models trained gradually

Day	Mimic Model	D	D+LC
20190622	MM20190622	96.57	94.48
20190703	MM20190622-0703	98.21	97.37
20190704	MM20190622-0704	95.77	94.01
20190705	MM20190622-0705	96.40	96.75

Major conclusions of this article is that multi-agent DQN incorporating appropriate extensions and following CTDE can scale up to the number of agents that operate in an airspace, providing qualitative solutions to resolving hostpots in heavy traffic days with mixtures of demand measures. Furthermore, experiments show that DQN can learn models that accumulate knowledge while being trained gradually in different scenarios, to a certain extent: these models do need further training towards refining the learned policies for the needs of scenarios to which they apply, balancing between training time and quality of solutions offered. Further work on this, aims to (a) show how these models can be learnt so as to accumulate new knowledge successfully, without deteriorating performance in scenarios to which they have already being trained, (b) improve their generalization abilities to reduce further the additional training needed before being applied to any scenario, (c) further improve our results by incorporating explicit inter-agent communication.

Regarding explainability, SGT prove to learn faithfully the DQN policy, while accumulating knowledge successfully as they are gradually being trained to subsequent DQN models: This allows building interpretable models incrementally, without reducing the fidelity of the models and the quality of explanations. Further work on this concerns using SGT in an online manner, exploiting the trained DQN incrementally, offering an inherently explainable deep reinforcement learning method.

However, explainability should be improved, both in providing explanation content, as well as in visualizing that content, so as to provide interpretations in more coarse granularity and at a large scale regarding spatial, temporal and societal dimensions, explaining the complexity involved in computing solutions without challenging the human cognitive abilities to the extent that we do now.

Table 14 Accuracy of MM20190622-0705 to all scenarios

Day	Mimic Model	D	D+LC
20190622	MM20190622-0705	92.76	93.00
20190703	MM20190622-0705	95.95	97.49
20190704	MM20190622-0705	92.55	92.08
20190705	MM20190622-0705	96.40	96.75
20190708	MM20190622-0705	94.18	92.11
20190717	MM20190622-0705	92.11	95.33

Table 15 Mean absolute error of SGT predictions

	Models			
Delay option	MM20190622	MM20190622 -0703	MM20190622 -0704	MM20190622 -0705
0	1.27	1.85	1.27	2.53
1	1.04	1.93	0.85	3.21
2	0.94	1.44	0.73	2.49
3	0.94	1.67	0.79	2.51
4	0.98	1.61	0.82	2.38
5	1.08	1.86	0.86	3.11
6	1.04	1.75	0.86	2.77
7	1.05	1.83	0.91	2.75
8	0.92	1.46	0.82	2.34
9	0.97	1.90	0.85	2.67
10	0.91	1.86	0.73	2.77

Acknowledgements This work has been supported by the TAPAS project, which has received funding from the SESAR Joint Undertaking under Grant Agreement No 892358 under European Union Horizon 2020 research and innovation programme.

References

1. Agogino AK, Tumer K (2012) A multiagent approach to managing air traffic flow. Auton Agents Multiagent Syst 24:1–25
2. Bazzan ALC (2009) Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. Auton Agent Multi-Agent Syst 18:342–375
3. Kuyer L, Whiteson S, Bakker B, Vlassis N (2008) Multiagent reinforcement learning for urban traffic control using coordination graphs. Mach Learn Knowl Discov Database:656–671
4. Tumer K, Agogino A (2007) Distributed agent-based air traffic flow management. International Conference on Autonomous Agents and Multiagent Systems (AAMAS '07)
5. Walraven E, Spaan MTJ, B.Bakker (2016) Traffic flow optimization: A reinforcement learning approach. Eng Appl Artif Intell 52:203–212
6. Dresner K, Stone P (2004) Multiagent traffic management: A reservation-based intersection control mechanism. International Conference on Autonomous Agents and Multiagent Systems (AAMAS '04)
7. Rosenthal RW (1973) A class of games processing pure-strategy nash equilibria. Int J Game Theory 2:65–67
8. Milchtaich I (2004) Social optimality and cooperation in nonatomic congestion games. J Econ Theory 114:56–87
9. Lipton ZC (2018) The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. Queue 16(3):31–57. https://doi.org/10.1145/3236386.3241340

10. Eurocontrol (2017) Performance review 2017
11. Eurocontrol (2018) Performance review 2018
12. Eurocontrol (2018) European aviation 2040: Challenges of growth
13. Eurocontrol (2018) European aviation in 2040 challenges of growth annex 1 flight forecast to 2040
14. Cook A (2016) European air traffic management. principles, practice and research. Rutledge, England
15. Kravaris T, Vouros G, Spatharis C, Blekas K, Chalkiadakis G (2017) Learning policies for resolving demand-capacity imbalances during pre-tactical air traffic management. Multiagent System Technologies - 15th German Conference (MATES '17), pp 238–255
16. Spatharis C, Kravaris T, Vouros GA, Blekas K, Cordero JMG (2018) Multiagent reinforcement learning methods for resolving demand - capacity imbalances. Digital Avionics Systems Conference (DASC'18)
17. Spatharis C, Kravaris T, Vouros GA, Blekas K, Chalkadiakis G, Garcia JMC, Fernãndez EC (2018) Multiagent reinforcement learning methods to resolve demand capacity balance problems. Hellenic A.I. Conference(SETN 2018)
18. Spatharis C, Bastas A, Kravaris T et al (2021) Hierarchical multiagent reinforcement learning schemes for air traffic management. Neural Comput Appl. https://doi.org/10.1007/s00521-021-05748-7
19. Kravaris T, Spatharis C, Bastas A, Vouros GA, Blekas K, Andrienko G, Andrienko N, Garcia JMC (2019) Resolving congestions in the air traffic management domain via multiagent reinforcement learning methods
20. Mukherjee P, Sen S, Airiau S (2008) Norm emergence under constrained interactions in diverse societies. Proceedings of the 7th international joint conference on autonomous agents and multiagent systems - vol 2, pp 779–786
21. Sugawara T (2014) Emergence of conventions for efficiently resolving conflicts in complex networks. 2014 IEEE/WIC/ACM international joint conferences on web intelligence (WI) and intelligent agent technologies (IAT), pp 222–229
22. Yu C, Zhang M, Ren F, Luo X (2013) Emergence of social norms through collective learning in networked agent societies. Proceedings of the 2013 international conference on autonomous agents and multi-agent systems, pp 475–482
23. Bowling M, Veloso M (2002) Multiagent learning using a variable learning rate. Artif Intell 136:215–250
24. Fudenberg D, Levine D (1998) The theory in learning in games
25. Shoham Y, Tennenholtz M (1997) On the emergence of social conventions: modeling, analysis, and simulations. Artif Intell 94:139–166
26. Vouros GA (2017) Learning conventions via social reinforcement learning in complex and open settings. Proceedings of the 16th conference on autonomous agents and multiagent systems, pp 455–463
27. Sen S, Airiau S (2007) Emergence of norms through social learning. Proceedings of the 20th international joint conference on artifical intelligence, pp 1507–1512
28. Airiau S, Sen S, Villatoro D (2014) Emergence of conventions through social learning. Auton Agent Multi-Agent Syst 28:779–804
29. Tan T, Bao F, Deng Y, Jin A, Dai Q, Wang J (2019) Cooperative deep reinforcement learning for large-scale traffic grid signal control. IEEE Trans Cybern:1–14
30. Agustín A, Alonso-Ayuso A, Escudero L, Pizarro-Romero C (2010) Mathematical optimizationg models for air traffic flow management: A review. Stud Inform Univ 8:141–184
31. Murça MCR (2018) Collaborative air traffic flow management: Incorporating airline preferences in rerouting decisions. J Air Transport Manag 71:97–107. https://doi.org/10.1016/j.jairtraman.2018.06.009
32. Agogino AK, Field M (2005) Multiagent reward analysis for learning in noisy domains. International conference on autonomous agents and multiagent systems (AAMAS '05), pp 81–88
33. Crespo A, Weigang L, Barros AD (2012) Reinforcement learning agents to tactical air traffic flow management. Int J Aviat Manag 1:145–161
34. Cruciol LBV, de Arruda A, Weigang L, Li L, Crespo A (2013) Reward functions for learning to control in air traffic flow management. Transp Res Part C: Emerg Technol 35:141–155
35. Jin F, Li Y, Sun S, Li H (2020) Forecasting air passenger demand with a new hybrid ensemble approach. J Air Transport Manag 83:101744
36. Rajendran S, Srinivas S, Grimshaw T (2021) Predicting demand for air taxi urban aviation services using machine learning algorithms. J Air Transport Manag 92:102043
37. Zahavy T, Ben-Zrihem N, Mannor S (2016) Graying the black box: Understanding dqns. In: Balcan MF, Weinberger KQ (eds) Proceedings of The 33rd international conference on machine learning, proceedings of machine learning research, vol 48. PMLR, New York, pp 1899–1908. http://proceedings.mlr.press/v48/zahavy16.html
38. Du W, Ding S (2021) A survey on multi-agent deep reinforcement learning: from the perspective of challenges and applications. Artif Intell Rev 54(5):3215–3238
39. Nguyen TT, Nguyen ND, Nahavandi S (2020) Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. IEEE Trans Cybern 50(9):3826–3839
40. Ding S, Zhao X, Xu X, Sun T, Jia W (2019) An effective asynchronous framework for small scale reinforcement learning problems. Appl Intell 49(12):4303–4318
41. Hernandez-Leal P, Kartal B, Taylor ME (2019) A survey and critique of multiagent deep reinforcement learning. Auton Agent Multi-Agent Syst 33(6):750–797
42. Kravaris T, Vouros GA (2021) Scalable deep multi-agent reinforcement learning. In: AAAI challenges and opportunities for multi-agent reinforcement learning (COMARL) Symposium. https://sites.google.com/view/comarl-aaai-2021/accepted-papers
43. Gupta JK et al (2017) Cooperative multi-agent control using deep reinforcement learning. In: AAMAS, pp 66–83
44. Sukhbaatar S et al (2016) Learning multiagent communication with backpropagation. In: NIPS, pp 2244–2252
45. Foerster J, Song F, Hughes E, Burch N, Dunning I, Whiteson S, Botvinick M, Bowling M (2019) Bayesian action decoder for deep multi-agent reinforcement learning. In: International conference on machine learning. PMLR, pp 1942–1951
46. Bouzy B (2017) Playing hanabi near-optimally. In: Advances in Computer Games. Springer, pp 51–62
47. Yang Y, Luo R, Li M, Zhou M, Zhang W, Wang J (2018) Mean field multi-agent reinforcement learning. In: International conference on machine learning. PMLR, pp 5571–5580
48. Jiang J, Lu Z (2018) Learning attentional communication for multi-agent cooperation. In: NIPS, pp 7254–7264
49. Jiang J, Dun C, Huang T, Lu Z (2019) Graph convolutional reinforcement learning. In: International conference on learning representations
50. Lin K et al (2018) Efficient large-scale fleet management via multi-agent deep reinforcement learning. In: 24th KDD

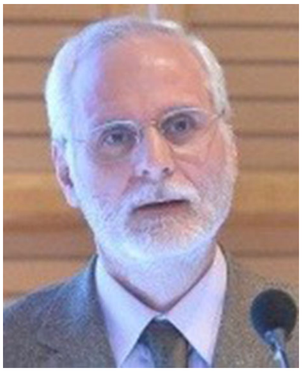
51. Nguyen DT, Kumar A, Lau HC (2017) Collective multiagent sequential decision making under uncertainty. In: Thirty-First AAAI conference on artificial intelligence
52. Nguyen DT, Kumar A, Lau HC (2017) Policy gradient with value function approximation for collective multiagent planning. (2017). Advances in Neural Information Processing Systems: Proceedings of NIPS, 4–9
53. Nguyen DT, Kumar A, Lau HC (2018) Credit assignment for collective multiagent rl with global rewards
54. Rusu AA, Colmenarejo SG, Gülçehre C, Desjardins G, Kirkpatrick J, Pascanu R, Mnih V, Kavukcuoglu K, Hadsell R (2016) Policy distillation. In: ICLR (Poster)
55. Degas A, Islam MR, Hurter C, Barua S, Rahman H, Poudel M, Ruscio D, Ahmed MU, Begum S, Rahman MA, Bonelli S, Cartocci G, Di Flumeri G, Borghini G, Babiloni F, Aricó P (2022) A survey on artificial intelligence (ai) and explainable ai in air traffic management: Current trends and development with future research trajectory. Appl Sci, 12(3). https://doi.org/10.3390/app12031295. https://www.mdpi.com/2076-3417/12/3/1295
56. Guestrin C, Lagoudakis M, Parr R (2002) Coordinated reinforcement learning. International Conference on Machine Learning (ICML ’02), 227–234
57. Mnih V et al (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
58. Tsitsiklis JN, Van Roy B (1997) An analysis of temporal-difference learning with function approximation. IEEE Transactions on Automatic Control 42(5):674–690
59. Van Hasselt H, Guez A, Silver D (2016) Deep reinforcement learning with double q-learning. In: Proceedings of the AAAI conference on artificial intelligence, vol 30
60. Hasselt H (2010) Double q-learning. Advances in neural information processing systems 23:2613–2621
61. Schaul T, Quan J, Antonoglou I, Silver D (2016) Prioritized experience replay. In: ICLR (Poster)
62. Behzadan V, Hsu W (2019) Analysis and improvement of adversarial training in dqn agents with adversarially-guided exploration (age). arXiv:1906.01119
63. Pinto L, Davidson J, Sukthankar R, Gupta A (2017) Robust adversarial reinforcement learning. In: International conference on machine learning. PMLR, pp 2817–2826
64. Gouk H, Pfahringer B, Frank E (2019) Stochastic gradient trees. In: Asian conference on machine learning. PMLR, pp 1094–1109
65. Andrienko N, Andrienko G, Fuchs G, Slingsby A, Turkay C, Wrobel S (2020) Visual analytics for data scientists. Springer International Publishing, Basingstoke
66. Andrienko N, Andrienko G, Miksch S, Schumann H, Wrobel S (2021) A theoretical model for pattern discovery in visual analytics. Visual Informatics 5(1):23–42. https://doi.org/10.1016/j.visinf.2020.12.002
67. Andrienko N, Andrienko G, Adilova L, Wrobel S (2022) Visual analytics for human-centered machine learning. IEEE Comput Graph Appl 42(1):123–133. https://doi.org/10.1109/MCG.2021.3130314
68. Andrienko G, Andrienko N, Cordero Garcia JM, Hecker D, Vouros G (2022) Supporting visual exploration of iterative job scheduling. IEEE Comput Graph Appl, 1–1. https://doi.org/10.1109/MCG.2022.3163437
69. Champougny T (2020) Ddr2 reference manual for general users. EUROCONTROL 2.9.7

Publisher’s note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Theocharis Kravaris is a PhD candidate, researching Multi-agent Deep Reinforcement Learning, at the Department of Digital Systems in the University of Piraeus, under the supervision of Prof. George Vouros. He graduated (2017) as a MSc. student at the Department of Informatics and Telecommunications of the National and Kapodistrian University of Athens with a Specialization of Information and Data Management, having received his bachelor’s degree (2015) in the same department, at the Track of Computer Systems and Applications. He became a member of the University of Piraeus’s AI Lab (2017) and participated in the DART project (SESAR project), until the end of the project. He worked on the Engage Catalyst project from June 2019 until May 2020, with the objective to develop algorithms for data-driven imitation of aircraft trajectories. He is currently working on TAPAS, a project which aims to develop explainable Machine Learning methods and apply them in the air traffic management domain as well as SIMBAD, a project aiming to advance the efficient and reliable evaluation of the performance of policies and practices that support the safety, efficiency and sustainability of Europe’s air transport system.

Konstantinos Lentzos holds a BSc in Mathematics, an MSc in Mathematics, all from the University of Athens, Greece, and he currently pursues his MSc thesis in Artificial Intelligence in University of Piraeus, Greece. He has worked as a data analyst, software engineer and fellow researcher. Currently he teaches data visualisation and data science courses in the American College of Greece His research interests include machine learning. deep learning, reinforcement learning and explainability.

Georgios Santipantakis (B.Sc, M.Sc, Ph.D) holds a BSc in Electrical Engineering, (minor Computer Science) (2003), Technological Educational Institute (T.E.I.) of Crete, an MSc in Information Management (2009) and a PhD in Knowledge Representation and Reasoning (2014) from the Department of Information & Communication Systems Engineering, University of Aegean, Greece. Currently he is a post-doctoral researcher at University of Piraeus and member of AI-Group. His Ph.D. research has been funded by IRAKLITOS II PhD Scholarship (Scholarship for research by the Greek Ministry of Education and the European Union, 2014), and a Honours Scholarship from Aegean University for his MSc studies. He has been co-instructor for the postgraduate courses "Knowledge Management and Reasoning" ("Health Care Management - Health Informatics", University of Athens) and "Web Technologies" ("e-Learning", University of Piraeus). He has participated in both national and international research programs. He has published in international journals (KAIS, IJAIT, ESWA), conferences and workshops (SETN, ICTAI, KEOD, WoMO, ESWC, Semantics, WIMS). He was co-author of the paper honored by the best paper award of SETN2014. His research interests include Distributed Reasoning in Description Logics, Knowledge Representation and Reasoning with Big Data, collaborative ontology evolution, Link Discovery and Linked Data processing.



Prof. George A. Vouros is a Professor in the Department of Digital Systems in the University of Piraeus. His published work and scientific interests are in the areas of Expert Systems (has developed 4 successful systems in critical and complex domains during 1990-1997), Ontologies & Semantic Integration of data, Agents and Multi-Agent Systems Reinforcement and Imitation Learning in multiagent and complex settings (e.g. in the aviation domain). He served/serves as program chair, chair and member of organizing committees of national and international conferences and as member of steering committees/boards of international conferences/workshops. He has given tutorials and keynote speeches in conferences and he has organized several workshops in well-known conferences. He served as guest editor in special issues in well-reputed journals. He is/was senior researcher in numerous EU-funded and National research projects.

Dr. Gennady Andrienko (www.geoanalitycs.net) is a lead scientist responsible for visual analytics research at Fraunhofer Institute for Intelligent Analysis and Information Systems and part-time professor at City University London. Gennady Andrienko was a paper chair of IEEE VAST conference (2015-2016) and associate editor of IEEE Transactions on Visualization and Computer Graphics (2012-2016), Information Visualization and International Journal of Cartography. Contact him at gennady.andrienko@iais.fraunhofer.de.

Dr. Natalia Andrienko, is a lead scientist at Fraunhofer Institute for Intelligent Analysis and Information Systems and part-time professor at City University London. Results of her research have been published in two monographs, "Exploratory Analysis of Spatial and Temporal Data: a Systematic Approach" (2006) and "Visual Analytics of Movement" (2013), and in a textbook "Visual Analytics for Data Scientists" (2020). Natalia Andrienko is an associate editor of IEEE Transactions on Visualization and Computer Graphics (2016-2020) and Visual Informatics. Contact her at natalia.andrienko@iais.fraunhofer.de.

Ian Crook is the Managing Director of ISA Software in Europe and Technical Director of ISA Software LLC in the USA. He has more than 30-years experience in the Air Traffic Management domain with a focus on the design and development of new operational concepts including validation, simulation and safety/performance assessment areas using fast-time, real-time and human in the loop gaming exercises for clients and partners across Europe, in the USA and in Asia. He is also an accomplished software architect and has been involved in the conception, design and development of many of ISA's ATFCM/ATM/ATC/Airport simulation and analysis products since founding the company in 1998. He has also worked on a number of European Commision, SESAR and US NextGen research projects since first becoming involved in the EC DG-TREN Episode 3 project (2000), and has co-authored many scientific papers for presentation in scientific journals and at ATM/ATFCM/Safety conferences around the world.

Jose Manuel Cordero Garcia is a Principal Researcher at CRIDA (R&D unit of the Spanish ANSP, ENAIRE) in Madrid, with extensive experience in the Air Traffic Management domain in the areas of operational performance monitoring and management, data analysis, and data-driven modelling. He has lead a large number of research projects, highlighting SESAR Performance Management since 2016. José Manuel has co-authored more than 50 peer-reviewed papers in journals and conferences, receiving two best paper awards. He also received the Jane' ATC Environment Award and Maverick Sustainability Award, recognizing achievements in green ATM concepts. Since August 2019, he serves as a member of the EUROCONTROL Performance Review Commission.

Enrique Iglesias Martinez is R&D Engineer at CRIDA. He is MsC Aeronautical Engineer by the Polytechnic University of Madrid. In 2019, he did a secondment at the Performance Review Unit (PRU) in Eurocontrol, Brussels, working in the refinement of performance indicators. At CRIDA, he is part of the ATM-DCB team and the link with PRU for ATM data provision. He is also responsible for the processing and integration into PERSEO tool of airport performance, dealing with data management and exploitation. With over 8 years of experience in the Air Traffic Management domain, he has also worked in some relevant validations under the scope of SESAR U-space projects. His project portfolio includes SESAR 1 and SESAR 2020, DART, datAcron, RETINA and DOMUS.