Learning Policies for Resolving Demand-Capacity Imbalances during Pre-tactical Air Traffic Management

Theocharis Kravaris[1], George A. Vouros[1], Christos Spatharis[2], Konstantinos Blekas[2], Georgios Chalkiadakis[3], and Jose Manuel Cordero Garcia[4]

University of Piraeus Research Centrer, Piraeus, Greece **

Abstract. In this work we propose and investigate the use of *collaborative reinforcement learning* methods for resolving demand-capacity imbalances during pre-tactical Air Traffic Management. By so doing, we also initiate the study of data-driven techniques for predicting multiple correlated aircraft trajectories; and, as such, respond to a need identified in contemporary research and practice in air-traffic management. Our simulations, designed based on real-world data, confirm the effectiveness of our methods in resolving the demand-capacity problem, even in the hardest of scenarios.

1 Introduction

The current Air Traffic Management (ATM) system worldwide is based on an time-based operations paradigm that leads to demand-capacity balancing (DCB) issues. These further impose limitations to the ATM system that are resolved via airspace management or flow management solutions, including regulations that generate delays (and costs) for the entire system. These demand-capacity imbalances are difficult to be predicted in pre-tactical phase (prior to operation) as the existing ATM information is not accurate enough during this phase.

With the aim of overcoming these ATM system drawbacks, different initiatives, notably SESAR in Europe[1] and Next Gen in the US[2], have promoted the transformation of the current ATM paradigm towards a new, *trajectory-based* operations (TBO) paradigm. In the future ATM system, the trajectory becomes the cornerstone upon which all the ATM capabilities will rely on. The trajectory life cycle describes the different stages from the trajectory planning, negotiation

** 1: Department of Digital Systems, University of Piraeus, Greece
2: Department of Computer Science & Engineering, University of Ioannina, Greece
3: School of Electrical and Computer Engineering, Technical University of Crete, Greece
4: CRIDA - Reference Center for Research, Development and Innovation in ATM, Madrid, Spain

[1] SESAR 2020, http://www.sesarju.eu/

[2] NextGen,https://www.faa.gov/nextgen/

and agreement, to the trajectory execution, amendment and modification. This life cycle also provides new opportunities in terms of both information quality and availability among ATM stakeholders, as it requires collaborative planning processes, before operations. The envisioned advanced decision support tools required for enabling future ATM capabilities will exploit trajectory information to provide optimised services to all ATM stakeholders—Airspace users, Air Navigation Service Providers, Network Manager, and so on.

The proposed transformation requires high-fidelity aircraft trajectory prediction capabilities, supporting the trajectory life cycle at all stages efficiently. This is also evidenced by the fact that improvements in trajectory prediction are fully aligned with FlightPath 2050[3] goals, in particular with those related to societal and market needs (with focus on improved, weather-independent arrival punctuality), protecting environment and energy supply, and ensuring safety and security. Single trajectory prediction refers to the process of predicting an individual trajectory considering it in isolation from the overall ATM system. Accounting for network effects and their implications on the execution of planned trajectories of individual flights requires considering interactions among these trajectories; moreover, it requires considering other operational conditions that influence the actual trajectory of any flight.

State-of-the-art techniques for predicting flights' trajectories, enable predictions based on specific physical models of aircrafts' movement, or on the exploitation of historical trajectory data that are obtained from surveillance systems (e.g., radar or ADS-B tracks) or directly from the aircraft (e.g., Quick Access Records). Two important drawbacks of such prediction methods are that *(a)* they are limited to single trajectory predictions, and *(b)* their prediction horizon is a short time one. Indeed, the trajectories are predicted one-by-one based on the information related to the individual flights, ignoring the expected traffic at the prediction time lapse. Consequently, the network effect resulting from the *interactions of multiple trajectories* is not considered at all, which may lead to huge prediction inaccuracies. This is due to the complex nature of the ATM system, which impacts the trajectory predictions in many different ways. Capturing aspects of that complexity, and being able to devise prediction methods that take the relevant information into account, would greatly improve the current trajectory prediction approaches.

Against this background, our main objective in this paper is to demonstrate how machine learning methods can help in refining single trajectory predictions (learned from surveillance data linked to weather data and other contextual information), considering cases where demand of airspace use exceeds capacity, resulting to *hotspots*. This is referred as the *Demand and Capacity Balance (DCB)* problem. In our work we study and determine the way trajectories are affected due to the influence of the surrounding traffic (i.e., considering interactions among individual predicted trajectories), taking into account an important aspect of ATM system complexity.

[3] "Flightpath 2050" European Commission. Available Online: http://ec.europa.eu/transport/modes/air/doc/flightpath2050.pdf.

Our overall, long-term goal is to deliver an understanding on the suitability of applying data-driven techniques for predicting single and multiple correlated aircraft trajectories. However, our focus in this article is on the DCB problem in Air Traffic Management, whose solution takes place during the so-called "flights' planning phase", during which the eventual conflicts resolutions adopted by air-traffic controllers in the actual flights are not taken into consideration. As such, our immediate objective is to predict delays that are applied to the flight plans, due to the demand and capacity imbalances occurring in hotspots.

To this end, this paper makes the following contributions:

- It formulates the DCB problem as an MDP.
- It proposes the use of specific *collaborative reinforcement learning* techniques for tackling this problem.
- It presents evaluation results in simulated, varying traffic conditions based on real-world data, showing the potential of our methods. All our methods managed to successfully resolve the DCB problem i.e., to produce schedules without any conflict seven in the hardest of our scenarios.

In the remainder of this paper we first state the operational context of our research in detail, motivate our work and state the problem to be solved (Section 2). Then in Section 3 we formulated the DCB problem in an MDP framework, and present collaborative reinforcement learning methods of choice. In Section 4 we present evaluation results for all our three methods. Section 5 presents relevant work; and finally, Section 6 concludes the paper.

2 The Air Traffic Management Operational Context and Motivation for Research

The operational scenarios for trajectory predictions considered in our research agenda assume that the process of predicting traffic happens at the planning phase (i.e., days before operation), as opposed to the tactical phase (i.e. in real-time during operation). The scenarios are considered to be developed in a specific geographical area (without affecting the generality of the solutions proposed), and interests of different stakeholders, such as Air Navigation Service Providers and airspace users, are taken into account: Air Navigation Service Providers require resolving the demand-capacity imbalances efficiently, while airspace users (e.g. airlines) aim to operate safely and efficiently without large delays.

Considering the ATM network effects and multiple trajectories prediction, our objective is to demonstrate how machine learning methods can help in refining single trajectory predictions considering cases where demand of airspace use exceeds capacity. Doing so, we aim to study and determine the way trajectories are affected due to the influence of the surrounding traffic. During the planning phase, conflict resolutions adopted by Air Traffic Controllers are not considered at all, so the resulting trajectories are not conflict-free.

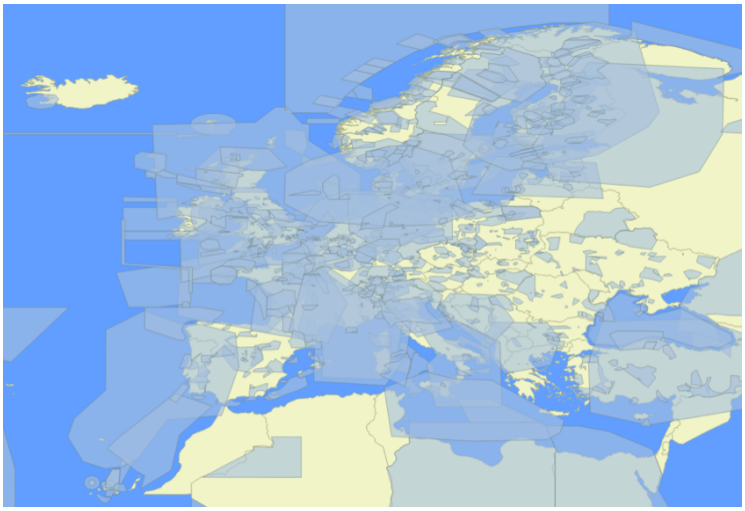


Fig. 1. Airlocks in 2D: Sectors are groups of adjacent airblocks.

2.1 The Demand-Capacity Balance Problem in ATM

The DCB problem (or process) considers two important types of objects in the ATM system: *aircraft trajectories* and *airspace sectors*.

Aircraft trajectories are series of spatio-temporal points of the generic form $(long_i, lat_i, alt_i, t_i)$, denoting the longitude, latitude and altitude, respectively, of the aircraft at a specific time point t_i. At the same time, *flight plans* are *intended trajectories*, which consist of events of flights crossing air blocks and sectors, and flying over specific waypoints. Each event specifies the element that is crossed (air block or sector), the entry and exit locations (coordinates + flight levels), and the entry and exit times, or the time that the flight will fly over a specific time. Other information such as estimated take-off time are specified, and, in case of delay, the calculated take-off time.

Sectors are air volumes segregating the airspace, each defined as a group of airlocks. These are specified by a geometry (the perimeter of their projection on earth) and their lowest and highest altitudes. As an example, Figure 1 depicts projections of airblocks above Europe. Airspace sectorization may be done in different ways, depending on sector configuration. Such a configuration determines the number of active (open) sectors. Only one sector configuration can be active at a time. Airspace sectorization changes frequently during the day, given different operational conditions and needs. This happens transparently for flights.

The *capacity of sectors* is of utmost importance: this quantity determines the maximum number of flights flying within a sector during a specific time interval.

The *demand* for each sector is the quantity that specifies the number of flights that co-occur (or predicted to occur) during a specific interval within a sector. Demand must not exceed sector capacity for any time interval. There

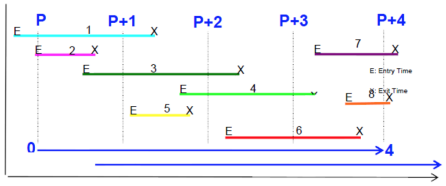


Fig. 2. Occupancy Step=1min., Duration=1min.

are different types of measures to monitor the demand evolution, with the most common ones being *Entry Rate* and *Occupancy Count*. In this work we consider Occupancy Count.

The Occupancy of a given sector is defined as the number of flights inside the sector during a selected period, referred as *Occupancy Counting Period*. In turn, this Occupancy Counting Period is defined as a picture of the sector occupancy taken every time step value along an interval of fixed duration: The Step value defines the time difference between two consecutive Occupancy Counting Periods. The Duration value defines the time difference between start and end times of each Occupancy Counting Period. For instance, considering the example in Figure 2 for a specific sector, the occupancy counts corresponding to the set of flights at different moments P with duration of 1min and step of 1min are: (a) At P: 1,2,3; (b) at P+1: 1,3,4,5; (c) at P+2: 3,4,6; and (d) at P+3: 4,6,7,8.

The DCB process is divided in three phases: Strategic, Planning and Tactical Phase. The overall objective is to optimise traffic flows according to air traffic control capacity while enabling airlines to operate safe and efficient flights.

Planning operations start as early as possible - sometimes more than one year in advance. Given that the objective is to protect air traffic control service of overload [5], this service is always looking for optimum traffic flow through a correct use of the capacity, guaranteed: safety, better use of capacity, equity, information sharing among stakeholders and fluency.

We consider the demand-capacity process during the *pre-tactical* phase. Pre-tactical flow management is applied at least six days prior to the day of operations, and consists of planning and coordination activities. This phase aims to compute the demand for the operations day, compare it with the predicted airspace capacities on that day, and make any necessary adjustments to the flight plans. Since our goal is trajectory prediction in a TBO environment, we consider individual predicted trajectories instead of flight plans, in order to determine the delay that should be imposed on them due to traffic. At this phase, trajectories are sent to the Network Manager who takes into account sector capacities to detect problematic areas. The main objective of this phase is to optimise efficiency and balance demand and capacity through an effective organisation of resources. In fact, DCB work today involves a collaborative decision making process among stakeholders, resulting to a corresponding Air Traffic Flow Control Management Daily Plan.

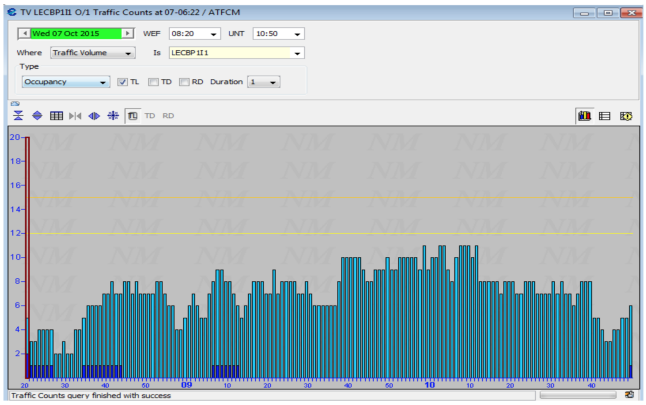


Fig. 3. Occupancy Indicator. The y axis represents the occupancy count, and the x axis time. Columns show occupancy counts, yellow line shows sustainable capacity and orange line shows the peak capacity

Tactical flow management takes place on the day of operations and involves considering, in real time, those events that affect the Air Traffic Flow Control Management Daily Plan and make the necessary modifications to it. This phase aims at refining the measures taken during the pre-tactical phases towards solving the demand -capacity imbalances that may appear. Tactical flow management is not within the scope of our work.

Figure 3 shows a snapshot of the Air Traffic Flow Control Management human-machine interface that is currently being used by the Network Manager, for supporting collaborative decision-making between all stakeholders: This snapshot shows the occupancy count of a specific sector in consecutive periods.

Concluding the above, our objective is to demonstrate how machine learning methods can help in trajectory forecasting when planned demand exceeds sectors capacity, taking into account interactions among trajectories, and thus traffic. In this case, regulations of type C (i.e. delays) are applied to the trajectories.

2.2 Problem Specification

Let there be N trajectories in $\mathcal{T}$ that must be executed over the airspace in a total time period of duration H (e.g. hours). The airspace consists of a set of sectors, denoted by $Sectors$. Time can be divided in intervals of duration Δt, equal to that of the Occupancy Counting Period.

As already defined above, each trajectory is a sequence of timed positions in airspace. This sequence can be exploited to compute the series of sectors that each flight crosses, together with the entry and exit time for each of these sectors. For the first (last) sector of the flight, i.e. where the departure (resp. arrival) airport resides, the entry (resp. exit) time is the departure (resp. arrival) time. However, there may exist flights that cross the airspace but do not depart and/or

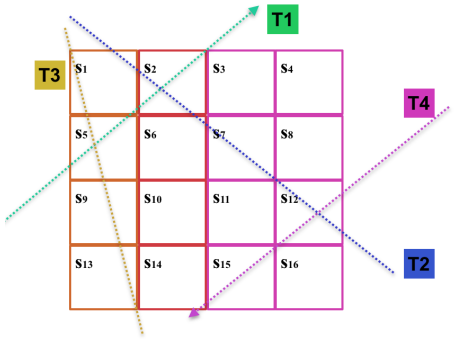


Fig. 4. Example of trajectories crossing sectors

arrive in any of the sectors of our airspace: In that case we only consider the entry and exit time of sectors within the airspace of our interest.

Thus, a trajectory T in $\mathcal{T}$ is a time series of elements of the form:

T=$\{(sector_1, entry_{t_1}, exit_{t_1})....(sector_m, entry_{t_m}, exit_{t_m})\}$,

where $sector_l \in Sectors, l = 1, ...m$.

For instance, considering the trajectories T_1 and T_2 in Figure 4, these are specified as follows:

$T_1 = \{(sector_5, 10{:}00,\ 10{:}20), (sector_2, 10{:}20,\ 10{:}45)\}$

$T_2 = \{(sector_1, 10{:}00,\ 10{:}05), (sector_2, 10{:}05,\ 10{:}15), (sector_7, 10{:}15,\ 10{:}25),$

$(sector_{12}, 10{:}25,\ 10{:}35)\}$

This information per trajectory suffices to measure the demand $D_{s_i,p}$ for each of the sectors $sector_i \in S$ in the airspace in any Occupancy Counting Period p of duration Δt.

Specifically, $D_{s_i,p} = |T_{s_i,p}|$, i.e. the number of trajectories in $T_{s_i,p}$, where

$T_{s_i,p} = \{T \in \mathcal{T} | T = (\ldots, (s_i, entry_{t_i}, exit_{t_i}), \ldots),$ *and the temporal interval* $[entry_{t_i}, exit_{t_i}]$ *overlaps with period* $p\}$

For instance, considering the trajectories T_1 and T_2 crossing the sector s_2 in Figure 4, it holds that $T_{sector_2,p} = \{T_1, T_2\}$, with p=[*10:10,10:15*].

The trajectories in $T_{sector_i,p}$ are defined to be *interacting trajectories* for the period p and the sector $sector_i$.

Each sector $sector_i \in S$ has a specific capacity C_{sector_i}. The aim is to resolve imbalances of sectors' demand and capacity: These are cases where $D_{sector_i,p} > C_{s_i}$, for any period p of duration Δt in H, in any $sector_i \in S$. Δt equals to the Occupancy Counting Period duration. We refer to these cases as *capacity violation* or *demand-capacity imbalance* cases, resulting to *hotspots*.

In case of capacity violation for a period p and sector $sector_i$, the interacting trajectories in $T_{sector_i,p}$ are defined as *hotspot-constituting trajectories*: one or more of these trajectories must be delayed in order to resolve the imbalance in $sector_i$.

Clearly, imposing delays to trajectories may propagate hotspots to a subsequent time period for the same and/or other sectors crossed by that trajectory: In any case, the sets of interacting trajectories in different periods and sectors may change, and thus, in case of demand-capacity imbalances, hotspot-constituting trajectories may change as well. This can be done in many ways, when different trajectories delay. Having said that, we must clarify that the only type of change in a trajectory that may be imposed by a regulation is "delay": i.e., shifting the entry and exit time for each sector by a specific amount of time. The sequence of sectors crossed is not affected.

Towards the agent-based formulation of the problem, we consider the following: Each agent A_i is specified to be the aircraft (instrument) performing a specific trajectory, in a specific date and time. Thus, we consider that agents and trajectories coincide in our case and we may interchangeably speak of agents A_i, trajectories T_i, or agents A_i executing trajectories T_i. Agents, as it will be specified, have own interests and preferences, although they are assumed collaborative, and take autonomous decisions on their delays: It must be noted that agents do not have communication and monitoring constraints given that imbalances are resolved at the planning phase, rather than during operation.

Therefore agents have to learn joint delays to be imposed to their trajectories w.r.t. the operational constraints concerning the capacity of sectors crossed by these trajectories. It must be noted that agents have conflicting preferences since they prefer to impose the smallest delay possible (preferably none) to their own trajectory, while also executing their planned trajectories safely and efficiently.

Agents with interacting trajectories are considered to be "peers" given that they have to jointly decide on their delays: The decision of one of them affects the others. This implies that agents form "neighbourhoods" of peers, taking also advantage of the inherent sparsity of the problem (e.g a flight crossing the north part of Spain, will never interact in any direct manner with a flight crossing the southest part of the Iberian Peninsula). However, as mentioned above, these neighbourhoods have to be updated when delays are imposed to trajectories, given that trajectories that did not interact prior to any delay may result to be interacting when a delay is imposed. Thus, a dynamic update of peers' neighbourhoods is necessary according to agents' decisions.

Given an agent A_i the traffic for that agent is determined to be the trajectories of all other agents forming its neighbourhood. More specifically:

Traffic(A_i)
$= \{T_j | T_j$ *is a trajectory that interacts with the trajectory* T_i *executed by* A_i *for any specific sector crossed by* T_i *and any time period within* H $\}$
$= \cup_{(sector,\cdot,\cdot)\in T_i,p} T_{sector,p}$

A society of agents $S = (\mathcal{T}, \mathcal{A}, \mathcal{E})$ is modelled as a graph with one vertex per agent A_i in $\mathcal{A}$ and any edge (A_i, A_j) in $\mathcal{E}$ connecting agents with interacting trajectories in $\mathcal{T}$. As pointed out above, the set of edges are dynamically updated by adding new edges when new interacting pairs of trajectories appear.

$N(A_i)$ denotes the neighbourhood of agent A_i, i.e. the set of agents connected to agent $A_i \in \mathcal{A}$ including also itself: These are the peers of A_i.

The options available in the inventory of any agent A_i for contributing to the resolution of hotspots may differ between agents: These, for agent A_i are $D_i \subseteq \{0, 1, 2, ..., MaxDelay_i\}$. These are ordered by the preference of agent A_i to any such option, according to the function $\gamma(i) : D_i \rightarrow \mathbb{R}$. We do not assume that agents in $\mathcal{A}-\{A_i\}$ have any information about $\gamma(i)$: This represents the situation where airlines set own options and preferences for delays even in different own flights, depending on different circumstances. However, we expect that the order of preferences should be decreasing from 0 to $MaxDelay_i$. In this paper we ran experiments assuming that $D_i = D_j$, and thus $MaxDelay_i = MaxDelay_j$, and $\gamma(i)(d) = \gamma(j)(d)$. This assumption does not affect the generality of the proposed methods, which may be applied to any other case. However, this issue requires further investigation for agents to reach optimal solutions.

Considering two peers A_i and $A_j \in N(i)-\{A_i\}$, agents must select among the sets of available options D_i and D_j respectively, so as to increase their expected payoff w.r.t. their preferences on options, and resolve the DCB problem.

This problem specification emphasises on the following problem aspects: *(a)* Agents need to coordinate their strategies (i.e. chosen options to impose delays) to execute their trajectories jointly with others, taking into account traffic, w.r.t. their preferences and operational constraints; *(b)* agents need to explore and discover how different combinations of delays affect the joint performance of their trajectories w.r.t. the DCB process, given that the way different trajectories do interact is not known beforehand (agents do not know the interacting trajectories that emerge due to own decisions and decisions of others, and of course they do not know whether these interactions result to hotspots i.e., demand-capacity imbalances); and*(c)* agents' preferences on the options available may vary depending on the trajectory performed, and are kept private.

3 Collaborative Reinforcement Learning Methods

3.1 The MDP Framework

Using the model of collaborative multiagent MDP framework [11], [6] we assume:

-The **society** of agents $S = (\mathcal{T}, \mathcal{A}, \mathcal{E})$.
-A **time step** $t = t_0, t_1, t_2, t_3, ..., t_{max}$, where $(t_{max} - t_0) = H$.
-A **local state** per agent A_i at time t, comprising state variables that correspond to (a) the delay imposed to the trajectory T_i, ranging to the sets of options assumed by A_i, and (b) the number of hotspots in which A_i is involved in (for any of the sectors and time periods). Such a state is denoted s_i^t. The **joint state** $s_{i,j}^t$ of agents A_i and A_j at time t is the tuple of the state variables for both agents. This is generalised for any subset of agents in the society. A **global state** s^t at time t is the tuple of all agents' local states.
-The **local strategy** for agent A_i at time t, denoted by str_i^t is the action that

A_i performs at that specific point: An action for any agent at any time point, in case the agent is still on ground, may be, either impose a delay or not. Thus, at each time point the agent has to take a binary decision. When the agent flies, then it just follows the trajectory. The location (i.e. sector) of that agent at any time point can be calculated by consulting its trajectory. The **joint strategy of a subset of agents** A of $\mathcal{A}$ executing their trajectories (for instance of $N(A_i)$) at time t, is a tuple of local strategies, denoted by str_A^t (e.g. $str_{N(A_i)}^t$). The set of all joint strategies for $A \subset \mathcal{A}$ is denoted $Strategy_A$. The **joint strategy** for all agents $\mathcal{A}$ at time t is denoted str^t.
-The **state transition function** gives the transition to the joint state s^{t+1} based on the joint strategy str^t taken in joint state s^t.
Formally $Tr : State \times Strategy \rightarrow State$. It must be noticed that although this transition function may be deterministic in settings with perfect knowledge about society dynamics, the state transition per agent is stochastic, given that no agent has a global view of the society, of the decisions of others, while its neighbourhood gets updated. Thus no agent can predict how the joint state can be affected in the next time step. Thus, for agent A_i this transition function is actually $Tr : State_i \times Strategy_{\{A_i\}} \times State_i \rightarrow [0, 1]$, denoting the transition probability $p(s_i^{t+1}|s_i^t, str_i^t)$.
-The **local reward** of agent A_i, denoted Rwd_i, is the reward that the agent gets by executing its own trajectory in a specific joint state of its peers in $N(A_i)$, thus $Traffic(A_i)$, according to the sectors' capacities, and the joint strategy of agents in $N(A_i)$. The **joint reward**, denoted by Rwd_A, for a set of peers A specifies the reward received by agents in A by executing their actions in their joint state, according to their joint strategy.

The joint reward Rwd_A for $A \subseteq \mathcal{A}$ depends on the number of hotspots occurring while the agents execute their trajectories according to their joint strategy str_A^t in their joint state s_A^t, i.e. according to their decided delays, and also according to their preferences on the chosen delays. Formally:

$$Rwd_A(s_A^t, str_A^t) = \lambda_1 * X(str_A^t, s_A^t) + \lambda_2 * D(str_A^t, s_A^t) \quad (1)$$

where, $X(str_A^t, s_A^t)$ is equal to the total number of hotspots in which agents in A are involved while executing their joint strategy in their joint state (i.e. according to the delays decided up to t), $D(str_A^t, s_A^t) : s_A \rightarrow \mathbb{R}$, is a function aggregating the preferences of agents on their chosen delays. The parameters λ_1 and λ_2 are used for balancing between the interests of different stakeholders towards reaching an optimum solution. Currently we have set $\lambda_1 = -100$ and λ_2 is a very small number close to zero: Methods are indeed proved to be very sensitive to preferences on delays although they do favour small delays, and this requires further investigation as part of our future work.

Thus, the reward received by any agent depends on (a) the sectors' capacity and the hotspots in which they participate, and on (b) their preferences on delays while performing their trajectories jointly.

A **(local) policy** of an agent A_i is a function $\pi_i : State_i \rightarrow Strategy_{\{A_i\}}$ that returns local strategies for any given local state, for A_i to execute its trajectory.

The objective for any agent in the society is to find an optimal policy π^* that maximises the expected discounted future return

$$V_i^*(s) = max_{\pi_i} E[\sum_{t=0}^{\infty} \delta^t Rwd_i(s_i^t, \pi_i(s_i^t))|\pi_i)] \quad (2)$$

for each state s_i, while executing its trajectory. $\delta \in [0, 1]$ is the discount factor.

This model assumes the Markov property, assuming also that rewards and transition probabilities are independent of time. Thus, the state next to state s is denoted by s' and it is independent of time.

3.2 Collaborative Q-Learning Methods

$Q-$functions, or action-value functions, represent the future discounted reward for a state s when deciding on a specific strategy str for that state and behaving optimally from then on. The optimal policy for agents in s is to jointly make the choice $argmax_c Q^*(s, str)$, maximizing expected future discounted reward.

The next paragraphs describe three collaborative reinforcement learning methods that take advantage of the problem structure (i.e. interactions among flights), considering that agents do not know the transition and reward model (model-free methods) and interact concurrently with all their peers.

Independent Reinforcement Learners (Ind-Colab-RL): The independent learners Q-learning variant proposed in [7] decomposes the global Q-function into a linear combination of local agent-dependent Q-functions. Each local Q_i is based on the local state and local strategy for agent A_i:

$Q(s, a) = \sum_{i=1}^{|N|} Q_i(s_i, str_i)$

Dependencies between agents, and thus the coordination graph, are defined according to the agents' society specified above. It must be pointed out that these dependencies may be updated by adding new ones while solving the problem. Each agent observes its local state variables.

A local Q_i is updated using the global temporal-difference error, the difference between the current global Q-value and the expected future discounted return for the experienced state transition, using

$$Q_i(s_i, str_i) := Q_i(s_i, str_i) + \alpha[Rwd(s_{N(A_i)}, str_{N(A_i)}) + \delta max'_a Q(s'_i, str_i^*) - Q(s_i, str_i)] \quad (3)$$

where, str_i^* is the best strategy known to the agent for the state s'_i. It must be noticed that instead of the global reward $Rwd(s, str)$ used in [7], we use the reward received by the agent, taking into account only the joint state and joint strategy of its neighbourhood.

Edge-Based Collaborative Reinforcement Learners (Ed-Colab-RL): This is a variant of the edge-based update sparse cooperative edge-based Q-learning method proposed in [8]. Given two peer agents performing their tasks, A_i and A_j, the $Q-$function is denoted succinctly $Q_{i,j}(s_{i,j}, str_{i,j})$, where $s_{i,j}$ with abuse of notation denotes the joint state related to the two agents, and $str_{i,j}$ denotes the joint strategy for the two agents. The sum of all these edge-specific $Q-$functions defines the global $Q-$function.

The update function in this case is as follows:

$$Q_{i,j}(s_{i,j}, str_{i,j}) = Q_{i,j}(s_{i,j}, str_{i,j}) + \\ \alpha(\frac{Rwd_i(s_i, str_i)}{N(A_i)} + \frac{Rwd_j(s_j, str_j)}{N(A_j)} + \delta Q_{i,j}(s'_{i,j}, str^*_{i,j}) - Q_{i,j}(s_{i,j}, str_{i,j})) \quad (4)$$

where, $str^*_{i,j}$ is the best joint strategy for agents A_i and A_j and for the joint state $s'_{i,j}$. In this case this is approximated using the *max-plus message-passing algorithm* [10]. Actually, given the society of agents (i.e. the coordination graph), in order to compute the optimal joint action str^*, each agent A_i repeatedly sends a message μ_{ij} to its neighbors $A_j \in N(A_i)$. The message μ_{ij} can be regarded as a local payoff function of agent A_j and is calculated as

$$\mu_{ij}(str_j) \; = \; max_{str_i}\{Q_i(str_i) + Q_{ij}(str_i, str_j) + \sum_{k \in N(A_i) - A_j} \mu_{ki}(str_i)\}, \quad (5)$$

The local Q-function for A_i is defined as in the Ind-Colab-RL update case above (formula (3)). Agents decide on their best local strategy by computing

$$str^*_i \qquad = \qquad argmax_{str_i}(f_i(str_i) \qquad + \qquad \sum_{j \in N(A_i)} \mu_{ji}(str_i)) \quad (6)$$

Agent-Based Collaborative Reinforcement Learners (Ag-Colab-RL): This is a variant of the agent-based update sparse cooperative edge-based Q-learning method proposed in [8]. As in $Ed-Colab-RL$ method, given two peer agents performing their tasks,A_i and A_j, the $Q-$function is denoted succinctly $Q_{i,j}(s_{i,j}, str_{i,j})$, where $s_{i,j}$ denotes the joint state related to the two agents, and $str_{i,j}$ denotes the joint strategy for the two agents. The update function is as follows:

$$Q_{i,j}(s_{i,j}, str_{i,j}) = Q_{i,j}(s_{i,j}, str_{i,j}) + \\ \alpha \sum_{k \in \{i,j\}} \frac{(Rwd_{i,j}(s_{i,j}, str_{i,j}) + \delta Q_k(s'_k, str^*_k) - Q_k(s_k, str_k))}{|N(A_k)|} \quad (7)$$

where, str^*_k is the best strategy for agent A_k in state s'_k, $k \in \{i, j\}$. Agents, compute their local Q-functions and their best local strategy as in the $Ed - Colab - RL$ method.

4 Experimental Results

We have performed a series of experiments in order to test and compare the efficiency of the three collaborative Q-learning methods to resolving the DCB problem in ATM. The efficiency is measured by means of the resulting number of hotspots, the mean delay achieved and the distribution of interacting flights in Occupancy Counting Periods – in conjunction to the number of learning periods needed for methods to compute policies. To this purpose, we create specific simulation scenarios of trajectories crossing an airspace. The scenarios are artificial, but correspond to typical and difficult cases in the real world, found in datasets provided by CRIDA, the Spanish Reference Centre for Research, Development, and Innovation in ATM. They have been used during this phase of our research in order to control the experimental settings and explore the potential of the proposed methods.

For the simulation we consider that the airspace comprises a grid of sectors, all having a specific capacity value (that could possibly differ from sector to sector). Table 1 presents the data used in producing the experimental cases and the parameter values used in all simulated runs.

Table 1. Parameter values used during the simulated experiments

Parameter	value
grid structure of sectors	4×4
capacity of sectors, C	$\in [4, 10]$
number of planes, N	100
Duration and Step of Occupancy Counting Period	6
total time period duration H	180
maximum delay	10

All three approaches follow an ϵ-greedy exploration strategy starting from probability 0.8, which is gradually reduced in subsequent rounds. However the *Ind-Colab-RL* differs from the other methods in that it initiates an ϵ-greedy exploitation phase for 1000 rounds with high probability, while in a subsequent phase of 1000 rounds, it does pure exploration. To evaluate the three approaches in cases of varying difficulty, we modify the *capacity* of sectors (C), and the number m of sectors that each flight crosses. Herein we report results only for the most hard cases in the grid considered, where $m \in [3, 4]$. For every capacity value $C \in [4, 10]$, we generated 50 random experimental cases. Figure 5 shows the mean value and the standard deviation of the final (after learning) number of hotspots, as well as the mean delay for all flights and for all experiments performed. According to the results and as shown in Fig. 5 (a), all methods demonstrated very similar behaviour wrt. hotspots' eradication, with $Ed - Colab - RL$ being slightly more effective compared to others: The x-axis in Fig. 5 (a) shows the capacity of each sector, while y-axis shows the number of hotspots when agents'

strategies converge. When the capacity of sectors was greater than or equal to 7 all methods reached the optimum policy for the hotspot criterion. However, an improvement in the 'mean delay' criterion is shown in Fig. 5 (b) concerning the edge-based and the agent-based collaborative RL approaches: x-axis in this figure shows the varying capacity of each sector, and the y-axis shows the mean delay achieved by each method. $Ind-Colab-RL$ shows the worst performance, while the performance of $Ed-Colab-RL$ is similar to that of $Ag-Colab-RL$, although the later is more consistent while the capacity of sectors increases. This confirms that the proposed multi-agent formulation provides a promising framework for tackling the DCB problem.

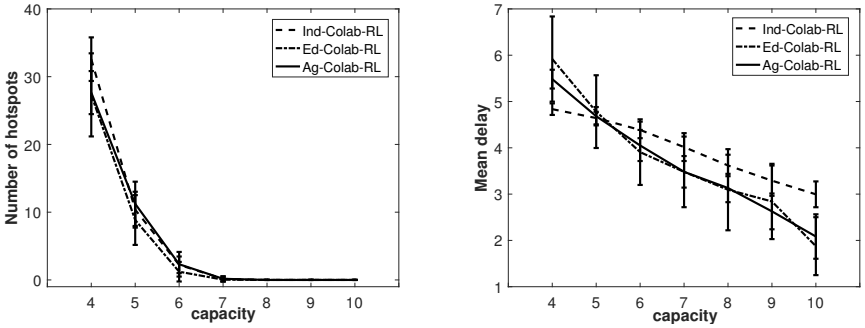


Fig. 5. Comparative results: Plots illustrate (a) the number of hotspots and (b) the mean delay estimated by each method in terms of various values of sectors' capacity (x-axis).

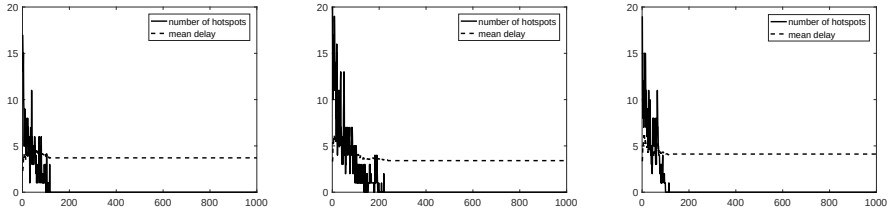


Fig. 6. Learning curves received by three methods in a setting considering sectors' capacity equal to 7. The x-axis shows the number of the learning episode, while the y-axis shows the number of hotspots and mean delay achieved in each episode.

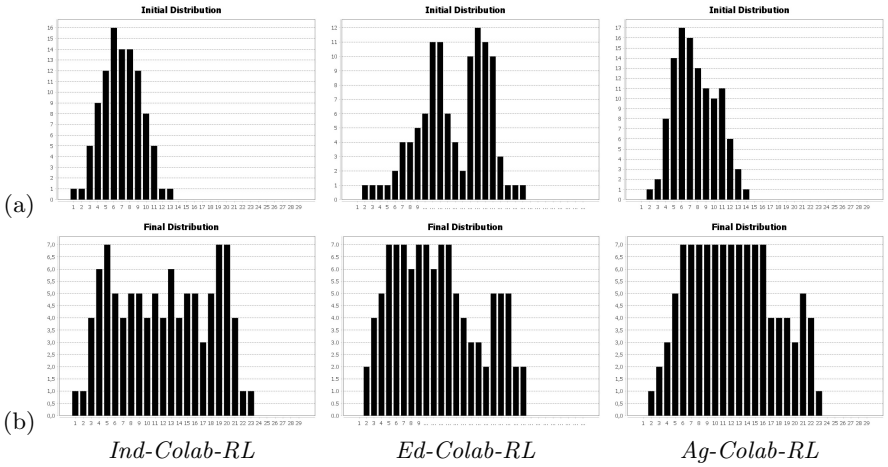
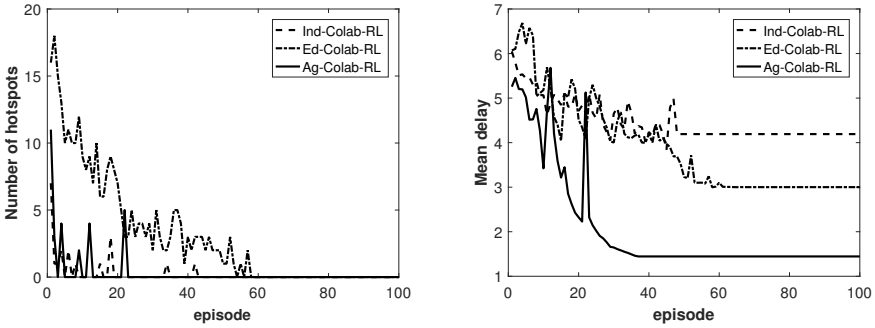


Fig. 7. An example of the distribution of interacting flights in Occupancy Counting Periods (a) initially and (b) as produced by three methods

Figure 6 illustrates an example of the received learning curves by each method, i.e. the number of hotspots and mean delay as estimated for 1000 episodes during learning (we set sector's capacity as $C = 7$ to all cases). For the $Ind-Collab-RL$ method, these episodes are from the pure exploration phase.

All methods were able to converge rapidly, achieving strategies with zero hotspots to any sector, and with flights' delay much less than the maximum acceptable delay (which was 10 in all experiments). Finally, in Figure 7 we present an example of the distribution of hotspots (y-axis) in terms of Occupancy Counting Periods in a number of 29 non-overlapping occupancy periods, each of duration equal to 6 time instants (e.g. 6 minutes). This was obtained by measuring the interacting flights to a specific sector in different periods: (a) at the beginning and (b) at the end of learning. As can be seen, our schemes manage to offer strategies with significantly reduced hotspots (zero in these cases, given that demand in any occupancy period is not greater than capacity).

Providing further evidence to the viability of the proposed methods, Figure 8 shows the learning curves received by the three methods in a setting where N=3000 and sectors' capacity C=20, while the remaining parameters are as specified in Table 1. In that figure the x-axis shows the number of the learning episode, while the y-axis shows the number of hotspots in each episode (Figure 8(a)) and the mean delay achieved per method (Figure 8(b)). As it is seen there, all methods converge fast, after only 60 episodes, resolving all imbalances. Specifically, the $Ag-Colab-RL$ method converges as fast as the $Ind-Colab-RL$, but in a solution where the mean delay is lower than those achieved by the other methods.



(a) Hotspots *(b) Mean Delay*

Fig. 8. Learning curves received by the three methods in a setting where N=3000 and sectors' capacity C=20. The x-axis shows the number of the learning episode, while the y-axis shows (a) the number of hotspots and (b) the mean delay achieved in each episode.

5 Related Work

Most works on agent-based modelling of the air traffic management system focus on the tactical phase, and mostly to the problem of avoiding collisions: These are mostly reactive approaches using probabilistic models [3] or geometric approaches [9]. For instance, following an agent-based approach, [2] and [12] propose decentralised methods for air traffic management application as well as for UAV collision avoidance. The first work proposes a negotiation approach for agents to find safe trajectories. Similarly in the second work agents, aiming to collision avoidance (tactical phase) following either an iterative p2p or a multi party collision avoidance method.

Using the Brahms multi-agent simulation framework, authors in [14] study the issues that affect the effectiveness of flow management in strategic planning. Although no decision-making or planning abilities are provided, the paper provides interesting insights for modelling the problem and addressing inefficiencies.

Closely to our aims, [1] propose multiagent reinforcement learning methods to reduce congestion through agents' local actions. Each agent may perform one of three actions: (a) setting separation between airplanes, (b) ordering ground delays or (c) performing reroutes. Agents are related to fixed points (sectors' entry points), while the hotspots are not guaranteed to be solved.

A recent work that provides Bayesian reinforcement learning (BRL) solutions for collaborative multiagent settings, is that of [13]. Like [8], the approach employs a variant of max-plus [10] for message-passing, but, crucially, it is able to extend single-agent and centralized multi-agent Bayesian RL methods [4] in collaborative settings by decomposing the coordination problem into *regional* sub-problems. In future work, we intend to explore the applicability of that paper's ideas in our problem domain.

6 Concluding remarks

This paper investigated a collaborative reinforcement learning framework for strategic planning of flight trajectories, with the aim of minimizing total conflicts and eliminating the effect of hotspots with minimum delays. The key aspect of the proposed scheme is the formulation of the DCB problem in the Air Traffic Management as a collaborative multiagent MDP framework where the aircrafts are treated as agents. Three multiagent RL schemes were studied, with their preliminary results being quite promising.

Our primary aim is to further extend our work in a variety of challenging issues. We intend to examine more systematically the generalization capabilities of the proposed RL-based multi-agents scheme to more complex environments and validate it in real operations.

This involves work in several interesting aspects: (a) Preparing datasets of flight plans in specific periods (e.g. days) of varying traffic. Historical data on flight plans do exist, including initial (unregulated) flight plans and their regulated versions per flight. (b) Exploiting historical data to train our methods and compute solutions using them, (c) tune/learn a reward model, and finally (d) compare delays imposed by our methods to those imposed by domain experts in real-life scenarios.

Of course the problem of resolving hotspots can be seen as a constraint optimisation problem (COP) and it is our aim to also compare the solutions produced by reinforcement learning methods to those produced by COP methods.

Another direction for future work is to introduce alternative joint Q-functions among agents taking into account geometric properties, and to exame different forms of the reward function.

Acknowledgements

This work is supported by the DART project, which has received funding from the SESAR Joint Undertaking under grant agreement No 699299 under European Unions Horizon 2020 research and innovation programme. For more details, please see the DART project's website, `http://www.dart-research.eu`

References

1. Agogino, A.K., Tumer, K.: A multiagent approach to managing air traffic flow. Autonomous Agents and Multi-Agent Systems 24(1), 1–25 (2012)
2. Albaker, B.M., Rahim, N.A.: Unmanned aircraft collision avoidance system using cooperative agent-based negotiation approach. Int. J. Simulation, Syst. Sci. Technol 11(4), 1–8 (2010)
3. Baek, K., Bang, H.: Ads-b based trajectory prediction and conflict detection for air traffic management. International Journal Aeronautical and Space Sciences 13(3), 377–385 (2012)
4. Chalkiadakis, G., Boutilier, C.: Coordination in Multiagent Reinforcement Learning: A Bayesian Approach. In: Proc. of AAMAS 2003. pp. 709–716 (2003)

5. Eurocontrol, h.t.f.a.c.m.: Air traffic flow and capacity management (atfcm) (2011)
6. Guestrin, C.E.: Planning Under Uncertainty in Complex Structured Environments. Ph.D. thesis, Stanford, CA, USA (2003), aAI3104233
7. Guestrin, C.G., Lagoudakis, M., Parr, R.: Coordinated reinforcement learning. In: In Proceedings of the ICML-2002 The Nineteenth International Conference on Machine Learning. pp. 227–234 (2002)
8. Kok, J.R., Vlassis, N.: Collaborative multiagent reinforcement learning by payoff propagation. J. Mach. Learn. Res. 7, 1789–1828 (Dec 2006), http://dl.acm.org/citation.cfm?id=1248547.1248612
9. Orefice, M., Di Vito, V., Corraro, F., Fasano, G., Accardo, D.: Aircraft conflict detection based on ads-b surveillance data. In: Metrology for Aerospace (MetroAeroSpace), 2014 IEEE. pp. 277–282. IEEE (2014)
10. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1988)
11. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, Inc., New York, NY, USA, 1st edn. (1994)
12. Sislak, D., Volf, P., Pechoucek, M.: Agent-based cooperative decentralized airplane-collision avoidance. IEEE Transactions on Intelligent Transportation Systems 12(1), 36–46 (2011)
13. Teacy, W.T.L., Chalkiadakis, G., Farinelli, A., Rogers, A., Jennings, N.R., McClean, S., Parr, G.: Decentralized bayesian reinforcement learning for online agent collaboration. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1. pp. 417–424. AAMAS '12, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2012), http://dl.acm.org/citation.cfm?id=2343576.2343636
14. Wolfe, S.R., Jarvis, P.A., Enomoto, F.Y., Sierhuis, M., van Putten, B.J.: A multi-agent simulation of collaborative air traffic flow management. In: Multi-Agent Systems for Traffic and Transportation Engineering, pp. 357–381. IGI Global (2009)