Multiagent Reinforcement Learning Methods for Resolving Demand – Capacity Imbalances

Theocharis Kravaris
University of Piraeus
Piraeus, Greece

Christos Spatharis
University of Piraeus Research Center,
Piraeus, Greece

University of Ioannina,
Ioannina, Greece

Konstantinos Blekas
University of Piraeus Research Center,
Piraeus, Greece

University of Ioannina,
Ioannina, Greece

George A. Vouros
University of Piraeus
Piraeus, Greece

georgev@unipi.gr

Jose Manuel Cordero Garcia
CRIDA
Madrid, Spain
jmcordero@crida.es

Abstract

In this article, we explore the computation of joint policies for autonomous agents, representing flights, to resolve congestions problems in the Air Traffic Management (ATM) domain in the context of Demand-Capacity Balance (DCB) process. We formalize the problem as a multi-agent Markov Decision Process (MDP) towards deciding flight ground delays to resolve imbalances, during the pre-tactical phase. To this end, we present and evaluate multi-agent reinforcement learning methods. An experimental study on real-world cases confirms the effectiveness of our approach.

Keywords—Demand-Capacity Problem, Multi-agent system, Reinforcement Learning

I. INTRODUCTION

Congestion problems, modeling situations where resources of a limited capacity have to be shared by multiple agents simultaneously, are ever present in the modern world. Most notably, congestion problems appear regularly in various traffic domains. It is of no surprise that they have drawn much attention in the AI and autonomous agents research for at least two decades now [3], and have been the focus of game theoretic models for much longer [14].

In the air-traffic management (ATM) domain, congestion problems arise naturally whenever demand of airspace use exceeds capacity, resulting to hotspots. This is known as the Demand and Capacity Balance (DCB) problem.

Specifically, the current ATM system worldwide is based on a time-based operations paradigm. As the system deals with an increasingly large number of flights, aiming to making efficient use of resources, this implies some limitations to the ATM system, often leading to DCB issues. These limitations 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.

Against this background, this article formalises the DCB problem as a multi-agent system, where agents, represenmting flights, aim to coordinate their joint actions with respect to operational constraints on the use of airspace. We consider planning air traffic management operations at the "pre-tactical" stage: Given air sectors' limited capacity, the issue at hand is to minimise scheduled flight arrival delays, and thus delay costs, while ensuring efficient utilisation of airspace. The problem is formulated as a multiagent MDP (MA-MDP). As part of this formulation, we devise a reward function that takes into account agents' contribution to DCB problems, ground delays and implied cost when agents deviate from their schedule. We then proceed to describe Collaborative Multiagent Reinforcement Learning (MARL) methods for learning joint policies towards resolving DCB problems, and explore the efficacy of these methods in real-world scenarios.

Our goal is to minimize the average ground delay per flight w.r.t. the number of flights with ground delay. In doing so, we aim to distribute ground delays among flights without penalizing a small number of them, and utilize efficiently the airspace so as to have an even distribution of demand to sectors in different periods.

Therefore, we consider only *ground delays* and subsequently we succinctly call these "delays".

The multiagent problem specification and corresponding proposed methods allow to assess ground delays to individual flights, always considering the *joint* effects of imposed delays to the evolution of demand. While agents represent flights, their environment comprises the airspace and other flights comprising "traffic". This is in contrast to regulating in a first-come-first-regulated basis - as it is the case today. Considering operational constraints for the joint performance of the tasks, the proposed multiagent methods support each individual agent to reconcile conflicting options (i.e. options creating hotspots) *jointly* with others and *jointly* decide about individual policies on delays, while possessing no information about the preferences and payoffs of others.

The proposed MARL methods are evaluated to real-world DCB problem cases, each one comprising flight plans for a specific day, above Spain. The data sources used to produce those cases include operational data regarding flight plans per day of operation, data regarding sector configurations at any given time, and reference values for the cost of strategic delay

to European airlines, currently used by SESAR 2020 Industrial Research.

An initial observation from the application of the MARL methods is that they, quite effectively, manage to provide solutions to the DCB problems, imposing delays that result to zero hotspots. By utilizing a variety of different metrics (such as the number of flights with delay, the average delay per flight with delay and average delay per flight, total delay time, and delay distributions) we provide evidence on the potential of the proposed methods to produce qualitative solutions: Indeed, results are quite significant since, in most of the cases the average delay per flight is reduced considerably, while a small percentage of flights have delay more than half an hour, while only a small percentage of flights get delay. To further assess the quality of solutions computed, these are compared to solutions provided by the Network Management organization (Eurocontrol, CFMU).

We envisage the work laid out in this paper to be seen as a first step towards devising agent-based methods for deciding on delay policies for correlated aircraft trajectories at the pre-tactical phase, answering the call for a transition to a Trajectory Based Operations (TBO) paradigm.

The structure of this paper is as follows: Section 2 provides a specification for the DCB problem, and Section 3 presents its formulation within an MA-MDP framework. Section 4 then presents the reinforcement learning methods proposed for solving the problem, and Section 5 presents experimental results. Section 6 presents related work and finally, Section 6 concludes the article outlining future research directions.

II. Problem Specification

As already pointed out, the current Air Traffic Management (ATM) system leads to demand-capacity imbalances.

With the aim of overcoming 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) one: 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 and agreement, to the trajectory execution, amendment and modification.

This life cycle requires collaborative planning processes, before operations: The envisioned advanced decision support tools will exploit trajectory information to provide optimised services to all ATM stakeholders. The proposed transformation requires high-fidelity aircraft trajectory prediction capabilities, supporting the trajectory life cycle at all stages efficiently.

The network effect resulting from the *interactions of multiple trajectories* is not considered at all by state-of-the-art techniques for assessing the impact of trafic to flights' trajectories wr.t. operational constraints. Considering flight trajectories in isolation from the overall ATM system may lead to inefficiencies to trajectory planning (due for instance to conflict resolution) and huge inaccuracies to assessing trajectory execution. Accounting for network effects and their implications on the joint execution of individual flights requires considering interactions among trajectories; moreover, it requires considering operational conditions that influence any flight. Capturing aspects of that complexity, and being able to devise methods that take the relevant information into account, would greatly improve the current trajectory prediction approaches. Our aim is to assess delays to be imposed to trajectories towards resolving DCB problems.

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

Sectors are air volumes segregating the airspace, each defined as a group of airblocks. These are specified by a geometry (the perimeter of their projection on earth) and their lowest and highest altitudes. Airspace sectorization may be done in different ways, depending on sectors' configuration, determining the 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 any time period of specific duration (e.g. in any 20' period).

The *demand* for each sector is the quantity that specifies the number of flights that co-occur during a time period within a sector. The duration of any such period is equal to the duration of period used for defining capacity. Demand must not exceed sector capacity for any time interval.

There are different types of measures to monitor the demand evolution, with the most common ones being Hourly Entry Count and Occupancy Count. In this work we consider Hourly Entry Count, as this is the one used by network managers at the pre-tactical stage.

The *Hourly Entry Count* (HEC) for a given sector is defined as the number of flights entering in the sector during a time period, referred to as an Entry Counting Period (or simply, counting period). HEC is defined to give a "picture" of the entry traffic, taken at every time "step" value along a period of fixed duration: The step value defines the time difference between two consecutive Entry Counting Periods. The *Duration* value defines the time difference between the start and end times of an Entry Counting Period. For example, for a 20 minutes step value and a 60 minutes duration value, *entry counts* correspond to pictures taken every 20 minutes, over a total duration of 60 minutes.

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. Casting them into a DCB resolution setting, trajectories may be seen as time series of events specifying the entry and exit locations (coordinates + flight levels) and the entry and exit times for the sectors crossed, or the time that the flight will fly over specific sectors. Thus, given that 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

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

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

each of these sectors. For the first (last) sector of the flight, where the departure (resp. arrival) airport resides, the entry (resp. exit) time is the departure (resp. arrival) time. For flights that cross the airspace but do not depart and/or arrive in any of the sectors of the airspace of interest, we only consider the entry and exit time for sectors within that airspace.

Therefore, we consider a finite set of discrete air sectors $\mathbf{R}= \{R_1, R_2,...\}$ segregating the airspace.

These sectors are related to a set of operational constraints, associated to their capacity, whose violation results to DCB (congestion) problems: These are cases where $D_{R,p} > C_R$, where p is a counting period of pre-defined duration d_t, $D_{R,p}$ is the demand for sector R during counting period p, and C_R is the capacity of the sector for any period of duration d_t equal to the counting period duration. These cases are *capacity violation* or *demand-capacity imbalance* cases, resulting to *hotspots*.

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

$T=\{ (R_1, entry_{t1}, exit_{t1}) .... (R_m, entry_m, exit_m) \}$,

where R_l, l=1,...m is a sector in the airspace.

This information per trajectory suffices to measure the demand for each of the sectors $R \in \mathbf{R}$ in the airspace, in any counting period p.

Specifically, the demand in sector R in period p is $D_{R,p}=|\mathbf{T}_{R,p}|$, i.e. the number of trajectories in $\mathbf{T}_{R,p}$, where

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

In case of hotspots, trajectories requiring the use of the sector R at the same period p, i.e. trajectories in $\mathbf{T}_{R,p}$, are defined to be *interacting trajectries* for p and R.

The overall objective of the DCB process at any phase of operations (Strategic, Planning and Tactical Phase) 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 [2], this service is always looking for optimum traffic flow through a correct use of the capacity, guaranteed: safety, better use of capacity, equity among flights and airlines, information sharing among stakeholders and fluency.

In this work we consider the demand-capacity process during the *pre-tactical* phase, assuming a trajectory-based operations environment with processes like RBT (Reference Business Trajectory) / SBT (Shared Business Trajectory) in place, enabling an enhanced accuracy of pre-tactical flight information. Pre-tactical flow management is applied at least six days prior to the day of operations, and consists of planning and coordination activities.

An agent A_i is the aircraft performing a specific flight trajectory (simply, trajectory), in a specific date and time. Thus, we consider that agents and trajectories coincide in this case and we may interchangeably speak of agents A_i, trajectories, flights 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 resolving hotspots: It must be noted that in the DCB problem agents do not have communication and monitoring restrictions, given that imbalances are resolved at the planning phase, rather than during operation.

To resolve hotspots, agents have several degrees of freedom: They may either change their trajectory, cross sectors other than the congested ones, or change their schedule of crossing the sectors. In this paper we consider only changing the schedule of crossing sectors by imposing *delays*: i.e., shifting the whole trajectory by a specific amount of time.

Now, the problem is about agents to execute their trajectories jointly, in an efficient and safe way, w.r.t. resources' operational constraints.

Specifically, in the DCB problem the goal is to

- minimize the average delay (ratio of total delay to the number of flights) w.r.t. the number of delayed flights; so as to
- distribute delays among flights without penalising a small number of them, and
- utilise efficiently the airspace so as to have an even distribution of demand to sectors in all counting periods within a total period of trajectories' execution $\mathbf{H}$.

To resolve a hotspot occuring in period p and sector R, a subset of interacting trajectories in $\mathbf{T}_{R,p}$ must be delayed. 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.

Clearly, imposing delays to trajectories may propagate hotspots to a subsequent time period for the same and/or other sectors crossed. Also, the sets of interacting tasks in different periods and sectors may change. This can be done in many different ways when imposing delays to flights, resulting to a dynamic setting for agents, where the sets of interacting trajectories do change according to agents' decisions.

Interacting agents contributing to hotspots are considered to be "peers" given that they have to jointly decide on their delays: The decision of one of them directly affects the others. This implies that agents form "neighbourhoods" of peers. Such neighbourhoods provide a way to take advantage of the spatial and temporal sparsity of the problem: For instance, in the DCB problem a flight crossing the northwest part of Spain in the morning, will never interact in any direct manner with a flight crossing the southeast part of the Iberian Peninsula at any time, or with an evening flight that crosses the northwest part of Spain. However, as mentioned above, these neighbourhoods have to be dynamicaly updated when delays are imposed to flights, given that trajectories that did not interact prior to any delay may result to be interacting when delays are imposed.

The *society of agents* ($\mathbf{A}$, $\mathbf{E}$) is modelled as a coordination graph with one vertex per agent A_i in $\mathbf{A}$ and any edge (A_i, A_j) in $\mathbf{E}$ connecting agents with interacting trajectories in $\mathbf{T}$. The set

of interacting agents, and thus edges, are dynamically updated when new interacting pairs of trajectories appear.

$\mathbf{N}(A_i)$ denotes the *neighbourhood* of agent A_i, i.e. the set of agents interacting with agent A_i in any period p and sector R, 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 in $\mathbf{D}_i = \{0,1,2,..., MaxDelay_i\}$. We consider that these may be ordered by the preference of agent A_i to any such option, according to the function $\gamma(i)$: $\mathbf{D}_i \rightarrow \Re$. We do not assume that agents in $\mathbf{A}$-$\{A_i\}$ have any information about $\gamma(i)$. This represents the situation where airlines set own options and preferences for delays even in different individual own flights, depending on operational circumstances, goals and constraints. We expect that the order of preferences should be decreasing from 0 to $MaxDelay_i$, although, with a different pace for different agents.

Problem statement: Considering any two peers A_i and A_j in the society (**A, E**), with A_j in $\mathbf{N}(A_i)$-$\{A_i\}$, these agents must select among the sets of available options $\mathbf{D}_i$ and $\mathbf{D}_j$ respectively, so as to increase their expected payoff w.r.t. their preferences on options $\gamma(i)$ and $\gamma(j)$, and resolve the DCB problem.

This problem specification emphasises on the following problem aspects:

Agents (i.e. individual flights) 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;

Agents (i.e. individual flights) need to jointly 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

Agents' preferences on the options available may vary depending on the trajectory performed, and are kept private;

There are multiple and interdependent hotspots that occur in the total period **H** and agents have to resolve them jointly;

The setting is highly dynamic given that hotspots change while agents choose their delay strategies, in ways that are unpredictable for agents.

III. MDP FORMULATION

According to the problem formulation stated above, and using the model of collaborative multi-agent MDP framework [7], we assume:

The *society of agents* (**A, E**), as described above.

A *time step* $t=t_0, t_1, t_2, t_3,..., t_{max}$, where $t_{max}- t_0=\mathbf{H}$.

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 executed by A_i, ranging to $\mathbf{D_i}=\{0,...MaxDelay_i\}$, and (b) the number of hotspots in which A_i is involved in. Such a local state is denoted s^t_i. The *joint state* $\mathbf{s}^t_{Ag}$ of a set of agents A_g at time t is the tuple of the state variables for all agents in A_g. 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 any subset A_g of agents is denoted $\mathbf{State}_{Ag}$, and the set of joint society states is denoted by **State**.

The *local strategy* for agent A_i at time t, denoted by str^t_i is the action that A_i performs at that specific time point: Such an action, in case the agent is still on ground, may be, either add to its total delay a delay for until the next time instant, or not. Thus, at each time point the agent has to take a binary decision. When the agent flies, then its strategy is determined by the intended trajectory.

The *joint strategy of a subset of agents* A_g of **A** executing their trajectories (e.g. of $\mathbf{N}(A_i)$) at time t, is a tuple of local strategies, denoted by $\mathbf{str}^t_{Ag}$ (e.g. $\mathbf{str}^t_{\mathbf{N}(Ai)}$). The *joint strategy* for all agents **A** at any time instant t is denoted $\mathbf{str}^t$.

The set of all joint strategies for any subset A_g of **A** is denoted $\mathbf{Strategy}_{Ag}$, and the set of joint society strategies is denoted by **Strategy.**

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

$$Tr: \mathbf{State} \times \mathbf{Strategy} \rightarrow \mathbf{State}.$$

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 sector configurations, 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

$$Tr: \mathbf{State}_{Ai} \times \mathbf{Strategy}_{Ai} \times \mathbf{State}_{Ai} \rightarrow [0,1],$$

denoting the transition probability $p(s^{t+1}_i | s^t_i, str^t_i)$.

The *local reward* of an agent A_i, denoted Rwd_{Ai}, is the reward that the agent gets by executing its own trajectory in a specific joint state of its peers in $\mathbf{N}(A_i)$, i.e. by considering any interacting trajectory, including itself. The *joint reward*, denoted by Rwd_{Ag} for a set of peers A_g specifies the reward received by agents in A_g by executing their actions in their joint state, according to their joint strategy.

The reward Rwd_{Ag} for and subset Ag of **A** depends on the participation (contribution) of agents in hotspots occurring while executing their trajectories according to their joint strategy $\mathbf{str}^t_{Ag}$ in their joint state $\mathbf{s}^t_{Ag}$, i.e. according to their decided delays. Formally:

$$Rwd_{Ag}(\mathbf{s}^t_{Ag}, \mathbf{str}^t_{Ag}) = C(\mathbf{s}^t_{Ag}, \mathbf{str}^t_{Ag}) - \lambda * DC(\mathbf{s}^t_{Ag}, \mathbf{str}^t_{Ag})$$

where,

$C(\mathbf{s}^t_{Ag}, \mathbf{str}^t_{Ag})$ is a function that depends on the participation of agents in hotspots while executing their joint strategy in their joint state, and $DC(\mathbf{s}^t_{Ag}, \mathbf{str}^t_{Ag})$ is a function aggregating agents' strategic delay costs.

The parameter λ is used for balancing between the number of hotspots and delays imposed to agents towards achieving the goal: Zero hotspots and the minimum possible delay per agent.

In the DCB problem, both functions $C(\mathbf{s}^t_{Ag}, \mathbf{str}^t_{Ag})$ and $DC(\mathbf{s}^t_{Ag}, \mathbf{str}^t_{Ag})$ represent strategic delay costs: We have chosen $C(\mathbf{s}^t_{Ag}, \mathbf{str}^t_{Ag})$ to depend on the total duration of the period in which agents fly over a congested sector. This is multiplied by 81 which is the average strategic delay cost per minute (in Euros) in Europe when 92% of the flights do not have delays [4]. If there is not any congestion, then this is a large positive constant that represents the reward agents get by not participating in any hotspot.

The actual form of $C(\mathbf{s}^t_{Ag}, \mathbf{str}^t_{Ag})$ is as follows:

$$C(\mathbf{s}^t_{Ag}, \mathbf{str}^t_{Ag}) = \begin{cases} -TDC*81 \; if \; TDC > 0 \\ PositiveReward \text{ if } TDC = 0 \end{cases}$$

where, TDC is the total duration in hotspots for agents in A_g. The first case holds when there are hotspots in which agents participate (thus, the total duration in hotspots, TDC, is above 0), while the second case holds when agents do not participate in hotspots.

The $DC(\mathbf{s}^t_{Ag}, \mathbf{str}^t_{Ag})$ component of the reward function corresponds to the *strategic delay cost* when flights delay at gate. In our implementation, this depends solely on the minutes of delay and the aircraft type, as specified in [4]. As such, the actual form of this function is as follows:

$DC(\mathbf{s}^t_{Ag}, \mathbf{str}^t_{Ag}) =$

$$-\sum_{A \in Ag} Delay_A * StrategicDelayCost(Delay_A, AircraftType(A))$$

where $Delay_A$ is the delay imposed to the agent A and $StrategicDelayCost$ is a function that returns the strategic delay cost given the aircraft type of agent A and its delay. Notice however that in the general case the function $DC(\mathbf{s}^t_{Ag}, \mathbf{str}^t_{Ag})$ *could be taking into account additional airline-specific strategic policies and considerations regarding flight delays*.

A (local) policy of an agent A_i is a function π_i: $\mathbf{State}_{Ai} \rightarrow \mathbf{Strategy}_{Ai}$ 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_i that maximises the expected discounted future return

$$V^*_{Ai}(s) = \max_{\pi i} E[\sum_{t=0}^{\infty} \delta^t * Rwd_{Ai}(s^t_{Ai}, \pi_i(s^t_{Ai})) | \pi_i]$$

for each state s^t_{Ai}, while A_i executes its trajectory. The discount factor δ, 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 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 strategy refers to.

IV. MARL Algorithms

We now describe the proposed MARL methods to deal with the multiagent joint DCB policy search problem. The key concept includes interactions between trajectories.

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 [16]. The optimal policy for any agent A in state s is the one maximizing the expected future discounted reward, i.e. $argmax_{str} Q(s,str)$.

In the next paragraphs we describe three alternative multiagent reinforcement learning approaches 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.

A. Independent Reinforcement Learners (IndLearners)

In the Independent Reinforcement Learners (IRL) framework, each agent learns its own policy independently from the others and treats other agents as part of the environment.

The independent Q-learning variant proposed in [6] decomposes the global Q-function into a linear combination of local agent-dependent Q-functions.

$$Q(\mathbf{s}, \mathbf{str}) = \sum_{i=1}^{|A|} Q_i(s, str)$$

Each local value, Q_i, for agent A_i is calculated according to the local state, s, and the local strategy, *str*.

The local value Q_i is updated according to the temporal-difference error, as follows:

$$Q_i(s, str) = Q_i(s, str) + \alpha[Rwd_i(s, str) + \delta \max_{str'} Q_i(s', str) - Q_i(s, str)]$$

It must be noted that instead of the global reward $Rwd(\mathbf{s}, \mathbf{str})$ used in [6], we use the reward received by the agent A_i, taking into account only the local state and local strategy. Furthermore, this method considers only local states and strategies, and it is in contrast to the approach of Coordinated Reinforcement Learning model proposed in [6], since that model needs agents to know the maximising *joint* action in the next state, the associated *maximal expected future return*, and needs to estimate the Q-value *in the global state*.

B. Edge-Based Collaborative Reinforcement Learners (EdgeBased)

This is a variant of the edge-based update sparse cooperative Q-learning method proposed in [8].

Multiple agents are jointly interacting with the environment.

Given two peer agents A_i and A_j connected by an edge in the coordination graph, the Q-function for these agents is denoted as $Q_{ij}(\mathbf{s}_{ij},\mathbf{str}_{ij})$, where $\mathbf{s}_{ij}$, with abuse of notation, denotes the joint state related to the set of the two agents A_i and A_j, and $\mathbf{str}_{ij}$ denotes the joint strategy for these two agents.

Half the sum of all edge-specific Q-functions defines the global Q-function, i.e.

$$Q(\mathbf{s}, \mathbf{str}) = 1/2 \sum_{i,j\in \mathbf{E}} Q_{ij}(\mathbf{s}_{ij}, \mathbf{str}_{ij})$$

The Q-learning update rule in this case is given by the following equation:

$$Q_{ij}(\mathbf{s}_{ij}, \mathbf{str}_{ij}) = Q_{ij}(\mathbf{s}_{ij}, \mathbf{str}_{ij}) + \alpha \left[\frac{Rwd_i(s_i,str_i)}{N(A_i)} + \frac{Rwd_j(s_j,str_j)}{N(A_j)} + \delta\ Q_{ij}(\mathbf{s'}_{ij}, \mathbf{str}^{*}_{ij}) - Q_{ij}(\mathbf{s}_{ij}, \mathbf{str}_{ij})\right],$$

where, $\mathbf{str^*}_{ij}$ in [8] is the *best* joint strategy for agents A_i and A_j for the joint state $\mathbf{s'}_{ij}$.

In our method, the strategy $\mathbf{str^*}_{ij}$ is the best strategy known by each agent and it is depicted directly from the agent's value function, $Q_i(s,str)$, which is calculated as the summation of local Qij values in its neighbourhood:

$str^*_i = \text{argmax}_{stri}\ Q_i(s_i,str_i)$, and

$Q_i(s_i,str_i) = \sum_{j\in N(A_i)} Q_{ij}(\mathbf{s}_{ij}, \mathbf{str}_{ij})$.

This is an approximation of the best action in each state, which is improved, as the agents learn. We experimentally found out that this approximation method offers comparable quality and considerable improvement in methods' computational efficiency than using computationally/communication intensive aproximation methods.

C. Agent-Based Collaborative Reinforcement Learners (AgentBased)

This is a variant of the agent-based update sparse cooperative Q-learning method proposed in [8]. As in EdgeBased method, given two peer agents performing their trajectories, A_i and A_j, their joint Q-function is denoted succinctly $Q_{ij}(\mathbf{s}_{ij},\mathbf{str}_{ij})$, where $\mathbf{s}_{ij}$ and $\mathbf{str}_{ij}$ denote the joint state and strategy, respectively, related to the two agents, as defined in the previous section. The update rule is then:

$$Q_{ij}(\mathbf{s}_{ij}, \mathbf{str}_{ij}) = Q_{ij}(\mathbf{s}_{ij}, \mathbf{str}_{ij}) + \alpha \left[\sum_{k\in\{i,j\}} \frac{Rwd_{ij}(\mathbf{s}_{ij},\mathbf{str}_{ij}) + \delta\ Q_k(s'_k,str^*_k) - Q_k(s_k,str_k)}{N(k)}\right]$$

where, str^*_k is the best known 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 EdgeBased method.

The main difference between the two previous methods and the IndLearners approach is that while the later take into account own states and strategies without sharing any information with others, the two collaborative approaches consider interacting agents, their joint policies and compute "joint" Q-values.

The two collaborative approaches differ on the way Q-values are updated: In particular, the EdgeBased update approach updates Q-values by propagating edge-specific temporal differences to the corresponding peer agents (i.e. those connected via the edge considered) and only along corresponding edges, considering local rewards of agents shared in their neighbourhood. On the otrher hand, the AgentBased update approach updates Q-values by propagating agents' temporal differences (i.e. those learned from their entire neighbourhood) to their peers along connecting edges, while it considers agents' joint reward. In doing so, in the agent-based update approach agents apply their strategies and learn from their entire neighbourhoods: Thus, the effects of their strategies, gathered from their neighbourhood are propagated to each of the peers along the edges.

V. Evaluation Results

A. Evaluation cases and metrics

To evaluate the proposed methods, we have constructed evaluation cases of varying difficulty. Each case corresponds to a specific day of 2016 above Spain and its difficulty has been determined by means of the number of flights involved, the average number of interacting flights per flight (which is translated to the average degree for each agent in the coordination graph, connecting that agent with its peers), the maximum delay imposed to flights for that day to resolve DCB problems according to CFMU data, the average delay for all regulated flights according to CFMU data, and the number of hotspots in relation to the number of flights participating in these hotspots.

The specific method used for constructing these evaluation cases is as follows:

The first step is to collect per chosen day all the Flight Plans as provided by the Spanish Operational Data source. Each Flight Plan may be associated to multiple Flight Plan Messages. Evaluation cases exploit only the plan specified in the last message arriving before takeoff. In order to identify it, the timestamp of the message arrival is compared to the Estimated Entry Date and Time to FIR (HFIR). Flight Plans spanning in two consecutive days (e.g. a flight could take off before and land after midnight) are considered for both days. In addition, the model of the aircraft is stored for the calculation of strategic delay costs. Finally, flights are distinguished between commercial and non-commercial, using their ID and the associated Flight Rules. Only commercial flights are subject to delays, although all flights participate to the computation of demand evolution.

After collecting the Flight Plans described above, we cross-check them with the CFMU dataset considering only flights the CFMU dataset records information about, while others are dropped. The cross identification is achieved by utilizing the ID of the flight, the departure and destination airport and the Initial Off-Block Time (IOBT). Moreover, delays imposed from the CFMU to resolve hotspots occurring inside the Spanish Airspace, are identified per flight.

Flight Plans specify a trajectory crossing air volumes. This sequence of events is exploited to compute the series of active sectors that each flight crosses- depending on the open airspace configurations, together with the entry and exit time for each of these sectors. For the first (last) sector of the flight, where the departure (resp. arrival) airport resides, the entry (resp. exit) time is the departure (resp. arrival) time. As already pointed out, there may exist flights that cross the airspace but do not depart and/or arrive in any of the sectors of our airspace: In that case we only consider the entry and exit time for the sectors within the airspace of our interest.

It must be noticed that given the delay imposed to a flight, sectors crossed by its trajectory may vary, due to the changing configurations. This may result to a number of alternative specifications of a single flight trajectory (each one crossing a different set of sectors), one for each possible delay.

Table 1 provides an overall description of the ten evaluation cases, identified by an integer and named by the day in which it occurred. Specifically, Table 1 specifies per case the following attributes (columns from left to right):

Number of flights: The number of flights for the corresponding day above Spain.

Average Degree in Coordination Graph (min/max): This indicates in average the number of interacting flights for each of the agents (flights) in each evaluation case. It is expected that as the coordination graph becomes more "dense", i.e. as the average degree per vertice (agent) increases, the problem becomes more computationally demanding. Min/Max indicates the minimum and the maximum degree reported in the coordination graph per evaluation case, while ignoring zeros.

MaxDelay (according to CFMU data): This is the MaxDelay for all flights: It is the maximum delay reported by CFMU data for that particular day.

Average Delay (according to CFMU data): This is the average delay for regulated flights reported by CFMU data for that particular day.

Number of regulated flights (type C – according to CFMU data): This is the number of flights with regulations of type C (i.e. delays due to DCB problems) reported by the CFMU for that particular day.

It must be noted that regulated flights by the Network Manager leave a large number of hotspots unresolved in any of the cases considered.

Number of hotspots (number of flights): This is the number of hotspots that exist in each case, together (in parenthesis) with the number of flights that participate to those hotspots (each flight may participate in different combinations of hotspots). This is also an indication of problems' difficulty: Of course, this difficulty may also depend on other factors such as the duration of flights to hotspots, the excess on capacity for these hotspots etc. It is not the purpose of this deliverable to delve into these issues, but we do need to indicate major differences among evaluation cases.

To measure the efficiency of the methods and the quality of solutions achieved, we have specified qualitative criteria (metrics) as follows:

TABLE I. EVALUATION CASES

Evaluation case	Number of Flights (Agents)	Average Degree in Coordination Graph (non-zero min/max)	MaxDelay (CFMU data)	Average Delay (CFMU data)	Num of Regulated Flights (type C) (CFMU data)	Num of hotspots (Num of flights)
1: Aug4	5544	6.41 (17-120)	66	12.41	179	33 (853)
2: Aug7	5868	8.03 (23-121)	112	17.54	475	42 (1104)
3: Aug10	5500	5.92 (19-125)	59	15.37	402	27 (759)
4: Aug13	6000	10.89 (22-105)	147	16.02	434	53 (1460)
5: Jul2	5572	6.39 (29-107)	80	17.89	521	29 (778)
6: Jul10	5824	9.98 (21-175)	175	17.35	305	51 (1320)
7: Jul12	5408	5.84 (21-95)	95	18.55	281	28 (820)
8: Jun5	5348	6.77 (20-111)	84	14.99	162	32 (803)
9: Sep2	5498	5.41 (21-112)	88	13.95	165	27 (754)
10: Sep3	5788	5.24 (18-77)	61	14.41	297	26 (783)

Learning curves of all methods showing the computational efficiency of the methods: These curves show per round of methods' application the average delay per flight in the evaluation case, as agents learn the ground-delay-policy to be applied. As algorithms converge to solutions, the number of hotspots should be reduced and eventually reach to zero, while the average delay should be reduced, signifying the computation of a solution. Therefore, the speed of reaching that point (zero hotspots) and the round at which methods stabilize agents' joint policy (remaining to zero hotspots and to a specific value for flights' average delay - without oscillating between non-solutions and/or solutions, and/or varying average delay values) signify the computational efficiency of the method to reaching solutions. Of course, in case a method cannot reach a solution for a specific case, it may converge to a joint policy that results to more than zero hotspots.

Number of flights with delay: This is the number of flights with at least 1min of delay, as decided by any of the methods.

Average delays: We report on (a) the average delay per flight with delay, as well as (b) the average delay per flight.

Distribution of delays to flights: To show how delays are distributed to flights, we provide histograms showing the number of flights to several ranges of delay duration (e.g. 10-29 min.) up to *MaxDelay*.

Evolution of demand: To further delve into the quality of solutions provided by the proposed methods, we provide for highly-demanded sectors the evolution of demand per counting period at the initial state (i.e. at the problem state) and the evolution of demand at the solution state (i.e. at the state where each method has converged to a joint policy for agents).

All measurements provided, result by averaging the measurements recorded by 5 independent experiments per case and method.

B. *Evaluation results*

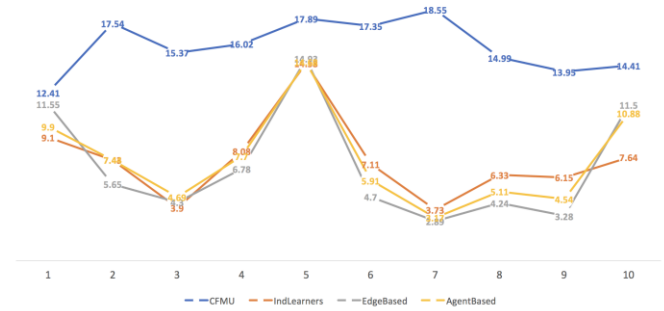
Regarding the delays imposed to flights according to CFMU data, as already pointed out, these do not resolve all DCB problems: I.e. even if we impose delays to flights according to the CFMU regulations we still have a large number of hotspots. Therefore, while the proposed methods do increase the number of flights with delay in all evaluation cases, the prescribed delays, applied to flights, result to 0 hotspots. Among the MARL methods, the EdgeBased method manages to have the less number of flights with delay among the rest of the methods in most of the cases. This is shown in Fig.1(b).

Regarding the average delay per flight with delay, as results reported in Fig.1(a) indicate, all methods manage to reduce considerably the CFMU average delay per flight with delay, in all cases. This is a remarkable result for all methods.

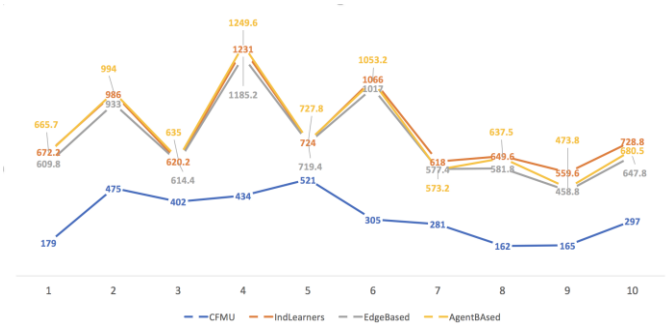
The EdgeBased approach seems to perform better than the other methods in all cases, except in three of them where IndLearners are more effective in reducing the delay. It should be noticed that in the majority of the cases the difference between the best and the second best average delays is quite large. This has not happened in the 5th case (Jul2), where all methods increase the average delay per flight with delay. Indeed, this case is considered to be the most difficult among all cases.

Considering the average delay per flight (actually the difference to the average delay per flight according to CFMU), as Fig.1(c) shows, in most of the evaluation cases, the proposed MARL methods managed to reduce the average delay compared to CFMU average delay per flight, while only in one of the methods the MARL methods have average delay per flight greater than 0.5 min compared to the CFMU average delay per flight: Among the methods, the EdgeBased shows to be more effective. It should be noted again that MARL methods manage to resolve all hotspots in any evaluation case, in contrast to what happens to CFMU regulations.

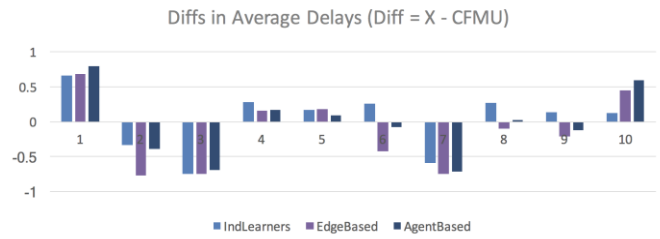
Learning curves in Fig.2 show that methods manage to converge effectively, given that until episode 7200 they are programmed to intervene exploitation with exploration. Due to space constraints, we provide results from two of the cases: Aug10 as being representative of all cases, and Jul2 which seems to be the most difficult among the cases. However, there are cases where methods can converge even earlier. Specifically, IndLearners manage to converge, or at least approximate effectively the convergence point, even earlier than episode 6000, except in one case – Jul2. All methods converge effectively to a solution after exploration round 7200, approaching the converge point. For Jul2 all methods converge quite late (i.e. after a large number of episodes).



(a)



(b)



(c)

Fig. 1. (a) Av. Delay per flight with delay (y-axis) per evaluation case (x-axis) and method, (b) Number of flights with delay (y-axis) per evaluation case (x-axis) and method, (c) Av. Delay per flight achieved by MARL method minus the Av.Delay per flight recorded in CFMU data per evaluation case (x-axis): High negative values are in favor of the MARL methods.

Regarding the fairness of methods and delving more into the quality of solutions: Each of the agents – in collaboration with its neighbors in the coordination graph (i.e. agents that correspond to interacting flights) - decides on own ground delay towards resolving the DCB problems in which it participates. According to the reward function that each agent evaluates independently from others, it aims to zero the hotspots in which it participates and minimize own delay. Therefore, as shown in Fig.3 for two of the cases (again, Aug10 and Jul2), the number of flights are reduced drastically as delays get larger. This happens in all cases. However, in the most difficult of the cases (e.g. Jul2) there are flights with large delays (e.g. more than 60').

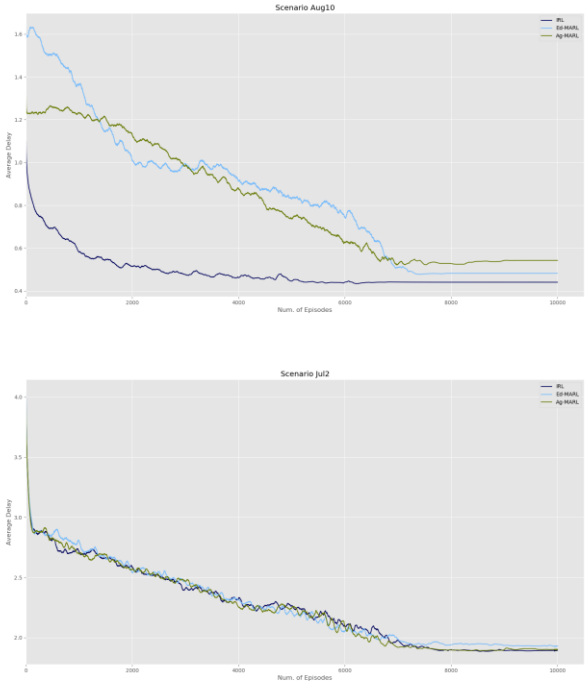


Fig. 2. Learning curves of all MARL methods for the Aug13 (top) and Jul2 (bottom) scenarios.

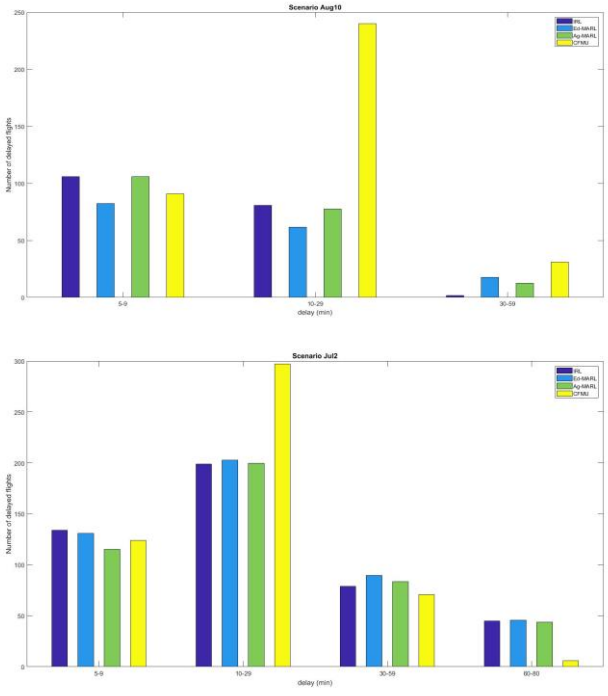


Fig. 3. Distribution of flights to ranges of delays for two evaluation cases: Aug13 (top), Jul2 (bottom).

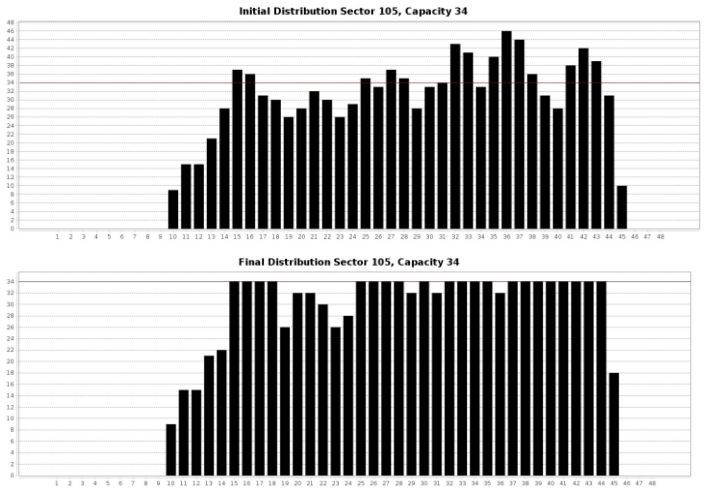


Fig. 4. Evolution of demand in Jul2 at the initial state (top- without imposing any delay), and at the solution state achieved by IndLearners for the most demanded sector in Jul2. The horizontal line shows the sector capacity.

Fig.3 provides further comparison of distribution of delays from agent-based methods and CFMU. It must be noticed that delays imposed by CFMU to the majority of the flights are within the range of 10 to 29 minutes. In few cases, CFMU imposes delays until 60 minutes to a considerable number of flights (e.g Aug10, Jul2).

Finally, Fig.4 shows the evolution of demand in different periods for the most demanded sector in the difficult case (Jul2) in (a) the initial problem, and (b) after imposing the ground delays decided by the IndLearners method. Results from the other methods and cases are similar (actually very close to those presented by the IndLearners) so we do not include them here. As Fig.4 shows, methods do "push" excess of capacity in subsequent periods within the same sector, or in other sectors (not shown here). This happens in small scale, i.e. solutions affect the demand for only 2 or 3 subsequent periods within the sector: This shows that delays imposed do not increase the workload per sector considerably, leaving much space for increasing further the demand, if this is also the case in the initial problem. It must be noted that interesting cases occuring (e.g. the one in period 36 in Fig.4) are difficult to explain due to complexity phenomena occuring among interacting flights and changing sector configurations.

VI. Related Work

Congestion problems have been studied extensively in game theory (see, e.g., [11][12][13]). Rosenthal [14] was the first to introduce and study congestion games, along with the existence of Nash equilibria. The use of multiagent systems techniques to tackle congestion problems in traffic and transprotation domains has been the focus of much work during the past two decades. Bazzan *et al.* [2] were among the first to frame congestion problems as a multiagent coordination problem, while Dresner and Stone [5] proposed a multiagent reservations-based traffic intersections control mechanism.

More recently, Agogino and Tumer [1] propose a multiagent framework for air-traffic control, and have evaluated their methods using real world airports' data. Agents

are assigned to specific ground locations throughout the airspace. The approach however does not guarantee to resolve hotspots and handles only up to two interdependent congestion instances.

Malialis *et al.* [10] use Q-learning to address multiagent congestion problems, while employing the idea of "resource abstraction": that is, they take into account the existence of abstract groups of congested resources, and provide increased punishment to agents when they are about to consume a congested resource. They evaluate their approach in road traffic simulation settings using up to 1000 agents, but do not tackle multiple congestion instances (e.g. hostspots), and do not use real-world data.

The only works we are aware of, which actually employ collaborative multi-agent RL methods, and which also apply them to resolve the pre-tactical DCB problem, is that of [9] and [15]. The work reported here extends these works by presenting extensive experimental results concerning the performance of MARL, providing evidence to the efficacy of the methods in real-world ATM settings.

VII. Concluding Remarks

In this paper we formulated the problem of resolving demand-capacity imbalances (DCB) in ATM as a coordination problem between agents controlling a multiagent MDP. We then proposed the use of MARL techniques to solve this problem, as a new paradigm for resolving DCB imbalances at the pre-tactical phase of operations. Our methods employ a novel, generic reward function that takes into account the agents' participation in hotspots, and also the strategic cost of delay.

We demonstrated the effectiveness of our methods by evaluating them on real-world scenarios encompassing thousands of agents in complex / dynamic settings. Experimental results in real-world problems show the potential of the proposed methods, in terms of efficiency (i.e. speed of convergence) and efficacy (in terms of quality of solutions achieved). In few words, collaborative MARL methods are promising to resolving real-world complex DCB problems in ATM.

More than that, we envisage the work laid out in this paper to be seen as a first step towards devising agent-based methods for prescribing the effect of traffic to correlated aircraft trajectories, contributing to the transition to a trajectory-based air-traffic management paradigm. This will hopefully help overcome the shortcomings of the currently used ATM paradigm, and as such, could in time allow commercial aircrafts "to fly their preferred trajectories without being constrained by airspace configurations"[1].

Acknowledgment

This work has been supported by the DART project, which has received funding from the SESAR Joint Undertaking under grant agreement No 699299 under European Union Horizon 2020 research and innovation programme; It has been partially funded by National Matching Funds 2017-2018 of the Greek Government, and more specifically by the General Secretariat for Research and Technology (GSRT), related to DART project. For more details, please see the DART project's website, http://www.dart-research.eu.

[1] https://www.sesarju.eu/vision

References

[1] Adrian K Agogino and Kagan Tumer. 2012. A multiagent approach to managing air traffic flow. Autonomous Agents and Multi-Agent Systems 24, 1 (2012), 1–25.

[2] Air Traffic Flow and Capacity Management (ATFCM). (2011). Eurocontrol, http://www.eurocontrol.int/articles/air-traffic-flow-and-capacity-management.

[3] Ana L. C. Bazzan, Joachim Wahle, and Franziska Klügl. 1999. Agents in Traffic Modelling - From Reactive to Social Behaviour. In KI-99: Advances in Artificial Intelligence, 23rd Annual German Conference on Artificial Intelligence, Bonn, Germany, September 13-15, 1999, Proceedings. 303–306.

[4] Andrew J Cook and Graham Tanner. 2015. European airline delay cost reference values. (2015). http://www.eurocontrol.int/publications/european-airline-delaycost- reference-values.

[5] K. Dresner and P. Stone. 2004. Multiagent traffic management: A reservation-based intersection control mechanism. In Proceedings of the 3rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS '04). 530–537.

[6] Carlos Guestrin, Michail Lagoudakis, and Ronald Parr. 2002. Coordinated Reinforcement Learning. In In Proceedings of the ICML-2002 The Nineteenth International Conference on Machine Learning. 227–234.

[7] Carlos Guestrin. 2003. Planning Under Uncertainty in Complex Structured Environments. Ph.D. Dissertation. Stanford, CA, USA. Advisor(s) Koller, Daphne. AAI3104233.

[8] Jelle R. Kok and Nikos Vlassis. 2006. Collaborative Multiagent Reinforcement Learning by Payoff Propagation. J. Mach. Learn. Res. 7 (Dec. 2006), 1789–1828. http://dl.acm.org/citation.cfm?id=1248547.1248612

[9] Theocharis Kravaris, et al. 2017. Learning Policies for Resolving Demand-Capacity Imbalances During Pre-tactical Air Traffic Management. In Multiagent System Technologies - 15th German Conference, MATES 2017, Leipzig, Germany, August 23-26, 2017, Proceedings. 238–255.

[10] Kleanthis Malialis, Sam Devlin, and Daniel Kudenko. 2016. Resource Abstraction for Reinforcement Learning in Multiagent Congestion Problems. In Proceedings of the 15th International Conference on Autonomous Agents and Multiagent System (AAMAS '16). 503–511.

[11] C. Meyers. Network flow problems and congestion games: complexity and approximation results. Ph.D. Dissertation, 2016, Cambridge, MA, USA.

[12] I. Milchtaich. Social Optimality and Cooperation in Nonatomic Congestion Games. Journal of Economic Theory 114 (2004), 56–87.

[13] Michal Penn, Maria Polukarov, and Moshe Tennenholtz. 2005. Congestion games with failures. In Proceedings 6th ACM Conference on Electronic Commerce (EC- 2005), Vancouver, BC, Canada, June 5-8, 2005. 259–268.

[14] R.W. Rosenthal. A Class of Games Possessing Pure-Strategy Nash Equilibria. International Journal of Game Theory 2 (1973), 65–67.

[15] Christos Spatharis et al. 2018. Multiagent Reinforcement Learning Methods to Resolve Demand Capacity Balance Problems, 10th Hellenic A.I. Conference, SETN 2018, Patras, Greece doi>10.1145/3200947.3201010

[16] Richard S Sutton and Andrew G Barto. 2014. Reinforcement learning: An introduction. MIT press Cambridge.