NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS & TELECOMMUNICATIONS

GRADUATE PROGRAM
INFORMATION AND DATA MANAGEMENT

POSTGRADUATE THESIS

Collaborative Reinforcement Learning for Resolving Hotspots in the Air Traffic Management Domain

Theocharis A. Kravaris

Supervisors: Panagiotis Stamatopoulos, Assistant Professor, NKUA
George Vouros, Professor, UPRC

ATHENS

JULY 2017

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Συνεργατική Ενισχυτική Μάθηση για την Επίλυση Συμφορήσεων στον τομέα Διαχείρισης Εναέριας Κυκλοφορίας

Θεοχάρης Α. Κράβαρης

Επιβλέποντες: Παναγιώτης Σταματόπουλος, Επίκουρος Καθηγητής, ΕΚΠΑ
Γεώργιος Βούρος, Καθηγητής, ΠΑΠΕΙ

ΑΘΗΝΑ

ΙΟΥΛΙΟΣ 2017

POSTGRADUATE THESIS

Collaborative Reinforcement Learning for Resolving Hotspots in the Air Traffic Management Domain

Theocharis A. Kravaris

A.M: M1457

Supervisors: **Panagiotis Stamatopoulos**, Assistant Professor, NKUA
George Vouros, Professor, UPRC

July 2017

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Συνεργατική Ενισχυτική Μάθηση για την Επίλυση Συμφορήσεων στον τομέα Διαχείρισης Εναέριας Κυκλοφορίας

Θεοχάρης Α. Κράβαρης
Α.Μ: Μ1457

Επιβλέποντες: Παναγιώτης Σταματόπουλος, Επίκουρος Καθηγητής, ΕΚΠΑ
Γεώργιος Βούρος, Καθηγητής, ΠΑΠΕΙ

Ιούλιος 2017

ABSTRACT

The objective of this thesis is to propose and investigate the use of collaborative reinforcement learning methods for resolving demand-capacity imbalances during pre-tactical Air Traffic Management. By so doing, it also initiates 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. This effectiveness is confirmed further by testing a real world scenario provided by CRIDA, the Spanish Reference Center for Research, Development, and Innovation in Air Traffic Management.

SUBJECT AREA: Artificial Intelligence, Machine Learning, Reinforcement Learning

KEYWORDS: Air Traffic Management, Demand Capacity Balance, Collaborative Multiagent Reinforcement Learning, Q-Learning

ACKNOWLEDGEMENTS

First of all, I would like to acknowledge my supervisor Prof. Panagiotis Stamatopoulos. His course Advanced Artificial Intelligence had directly contributed in furthering my interest in AI and assisted me in selecting this topic. I would also, like to acknowledge Prof. George Vouros, for his invaluable time and effort throughout the process of this thesis. Moreover, I would like to acknowledge Prof. Konstantinos Blekas and Prof. Georgios Chalkiadakis, whose insight assisted me during this project and MSc Student Christos Spatharis, for his valuable assistance in some crucial technical points. Finally, I would like to acknowledge the DART Project[1], that supported this thesis.

[1]DART has received funding from the SESAR Joint Undertaking under grant agreement No 699299 under European Unions Horizon 2020 research and innovation programme. DART project's website: http://www.dart-research.eu

Contents

List of Figures

List of Tables

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[2] and Next Gen in the US[3], 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 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[4] 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

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

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

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

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 thesis 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.

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 thesis 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 [2].

To this end, this thesis incorporates the following:

- 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 conflicts even in the hardest of our scenarios.

The rest of this thesis is structured as follows. In Chapter 2 we present the Demand Capacity Problem in ATM, the operational context and basic terminology. We also describe the problem and the Data Sources that should be exploited during the process. Chapter 3 is dedicated to the preliminaries, describing the theoretical background of MDPs and Reinforcement Learning. Chapter 4 gives a detailed problem specification. Chapter 5 presents the collaborative Reinforcement Learning algorithms used for the experiments. Chapter 6 contains the experimental evaluation. Chapter 7 presents notable work related to this thesis. Finally, in Chapter 8 there are some conclusions and propositions for future extensions of this work.

2. THE DEMAND-CAPACITY BALANCE PROBLEM IN ATM

2.1. Operational Context and Basic Terminology

Air Navigation is the combination of procedures and techniques that make possible an aircraft to fly from an origin to a destination. It is formed by many services supporting this purpose. Each country has mainly the same organization of such services, following the ICAO rules for air navigation [3] [4].

Air Navigation System is divided in three mainly services.

The first one is the **Aeronautical Information Service** (AIS). AIS is provided by Aeronautical Information Division to all users requesting it. AIS provides the necessary information so as to ensure that aeronautical operations are developed with safety, regularity, economy and efficiency. All the information is made public and distributed by the Air Navigation Service Provider of each country [5].

Meteorological Services contribute towards the safety, regularity and efficiency of international air navigation by the provision of timely and accurate weather information. It will be apparent that aircrew must be able to access accurate weather information when planning their flight and given the changing nature of the earth's weather patterns this information will need to be updated as necessary ensuring that a planned flight can be completed safely [6]. This is achieved by providing necessary meteorological information to aircraft operators, flight crew, air traffic services units and airport management through network of international communication systems which ensures close liaison between all stakeholders.

Air Traffic Management primarily consists of three distinct activities:

- Airspace Service Management (ASM): This service is responsible for airspace's planning and management. The main objective of this service is to allow safety, efficient and effective aircraft's operations. The service works in these main aspects: to build an airspace structure, to maintain airways (aircraft's routes) and to coordinate civil and military activity.
- Air Traffic Control: It's the process by which aircrafts are safely separated as they fly and at the airports where they land and take off. Tower control at airports is a familiar concept regarding air traffic control, but aircrafts are also separated as they fly en route; Europe has many large Air Traffic Control Centers which guide aircrafts to and from terminal areas around airports [4].
- Air Traffic Flow Management: It is an activity that is done before flights take place. Any aircraft using air traffic control, from a business aeroplane to an airliner, files a

flight plan and sends it to a central repository. All flight plans for flying into, out of and within Europe are analysed and computed [5].

Europe has a complex airspace, where 30.000 aircrafts usually overfly its sky. Therefore, it is one of the airspaces with most activity in the world. ATFCM service appears in ninety's, where European airspace has a huge lack of capacity taken into account the growth of demand. For this reason, a service available to handle capacity and demand balancing appeared early in ninety's [7] [8] [9]. The objective is to optimize 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: Air traffic forecasts issued are consolidated by the aviation industry and the capacity plans issued by the Air Traffic Control Centers and airports. Also operational scenarios to anticipate specific events which may cause congestion (such as sporting events, Christmas skiing or summer holiday traffic) are specified. Eventually, in case of an unforeseen event with major impact on traffic, a coordinated response to the crisis is organized. Given that the objective is to protect ATC service of overload, [9] 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. Coordination between actors in the system is necessary. The main actors involved are:

- Airline Operators (AO): Airlines must be informed of the regulations that are applied to their flights.
- Network Manager (NM): Central Position placed on Eurocontrol that is in charge of network monitoring in order to propose regulations to FMPs. Once these regulations are approved, these are applied to the flights affected.
- Flow Management Position (FMP): Local position placed at Airspace Control Centre (ACC) level that is in charge of network monitoring in order to approve the necessary regulations proposed by the NM.
- Airport Operators (AOP): Airports are the places where regulations are applied to specific flights. Operators must be informed of applied regulations to the flights while are still on the ground.

The Demand and Capacity Balancing (DCB) process is organized mainly in three phases, depending on the lookahead time: strategic, planning and tactical. The operational scenarios for trajectory predictions considered in this thesis (and as part of DART agenda of research) 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), and aims at improving the predictability of the traffic count within an airspace volume. 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 (ANSP) and airspace users, are taken into account: Air Navigation Service Providers (ANSP) 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.

2.2. Problem Description

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 2.1 (provided by Dr.Giorgos Santipantakis) 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.

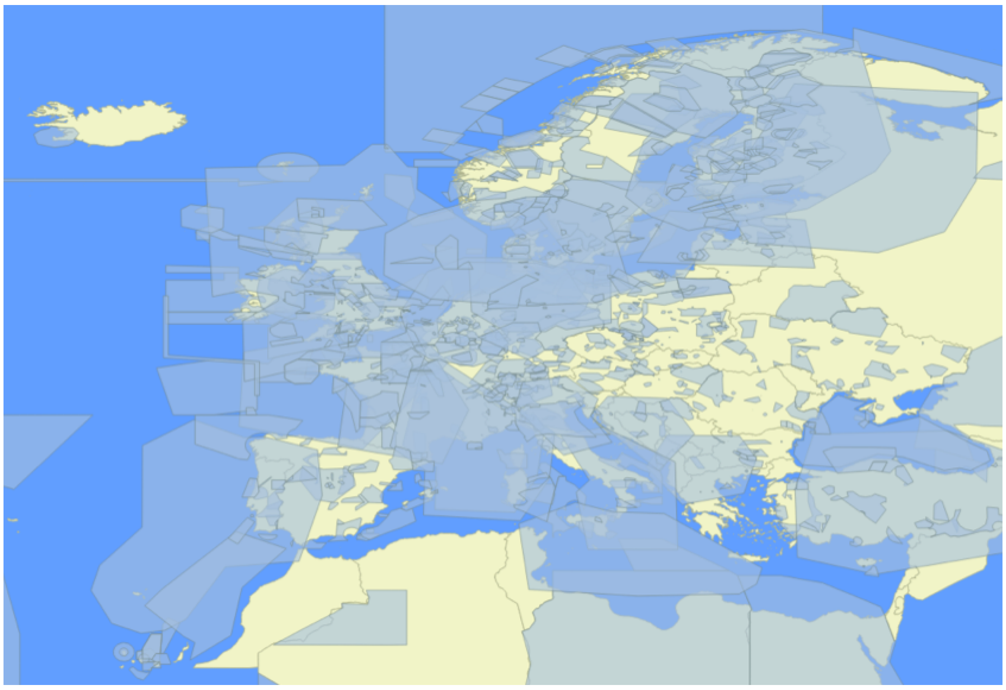


Figure 2.1: Airlocks in 2D: Sectors are groups of adjacent airblocks.

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 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 2.2 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.

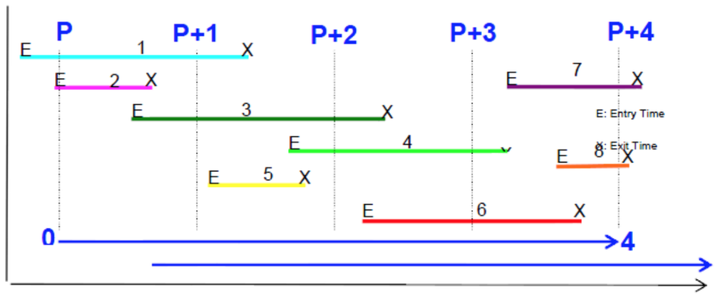


Figure 2.2: Occupancy Step=1min., Duration=1min.

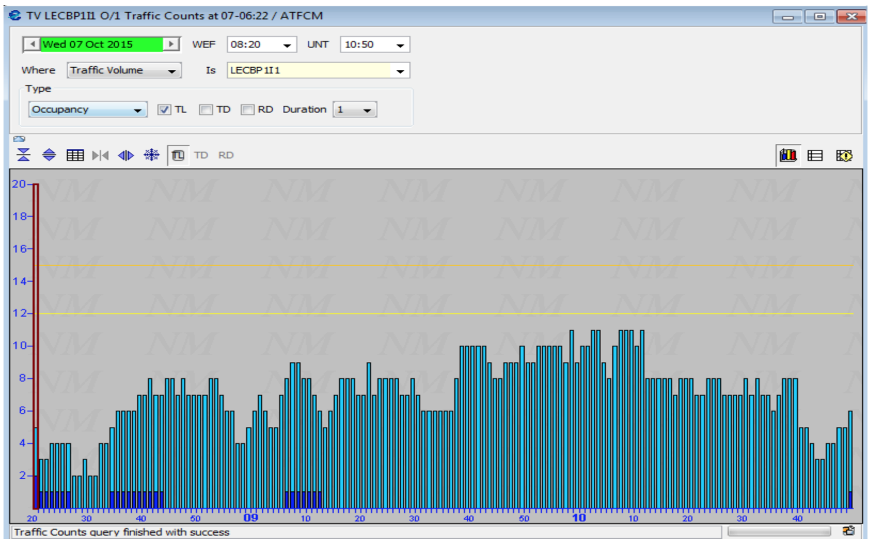
The DCB process is divided in three phases: Strategic, Planning and Tactical Phase. The overall objective is to optimise traffic flows according to air traffic control capacity while enabling airlines to operate safe and efficient flights.

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 [10], 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.

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 2.3 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.


TV LECBP1I1 O/1 Traffic Counts at 07-06:22 / ATFCM

Wed 07 Oct 2015 WEF 08:20 UNT 10:50

Where Traffic Volume Is LECBP1I1

Type Occupancy ☑ TL ☐ TD ☐ RD Duration 1

Traffic Counts query finished with success

Figure 2.3: 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

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.

2.3. Data Sources

This section presents the data sources exploited for the experimental evaluation of the algorithms implemented. Before continuing, I would like to acknowledge DART and CRIDA for providing these data sources, which are confidential and cannot be provided as part of this thesis.

The data sources include the following vital information:

- Tables with all the flight plan messages, from which we can extract flight plans.
- Tables which contain sectorizarion information, such as sectors' capacities.
- Tables to specify which sector configuration is active each given moment.

In order to create a scenario based on the data sources, the initial task was to collect the flight plans. As previously stated, flight plans consist of air volumes (or sectors) crossed by a flight, as well as time stamps for entering and exiting each air block.

To accomplish that task, we needed to filter flight plan messages. In addition, due to the fact that dozens of flight plan messages could correspond to each flight, we needed to assure that we used the initial flight plan messages, meaning the ones that occur before any kind of delay was applied. After collecting all the initial flight plans for the duration needed, specifically for a single day, we could store the required information i.e.the air volumes and time stamps concerning each flight plan. Figure 2.4 shows the air traffic in all the volumes crossed during the 12th of January 2016. Figure 2.5 shows the flights that cross the volume with the most traffic that day.

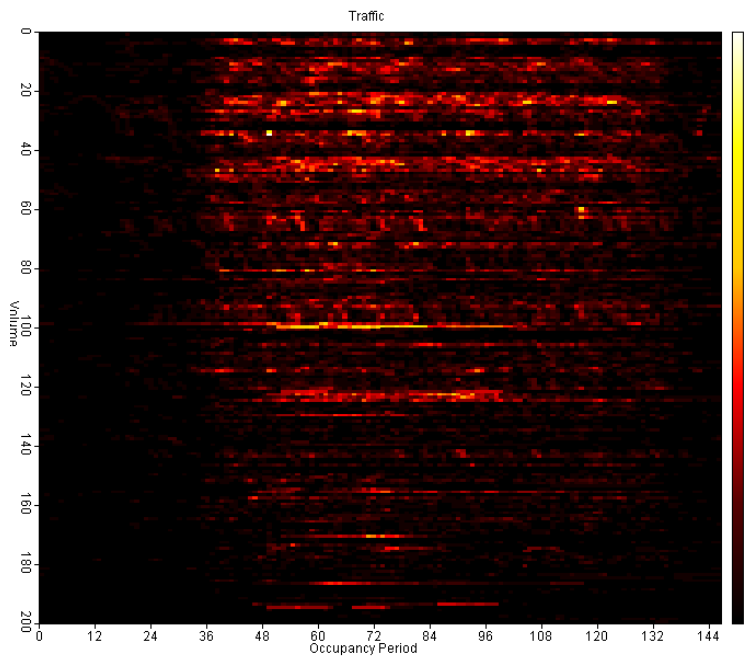


Figure 2.4: Air Traffic during 12/01/2016. Darker colors indicate lower traffic, as colors become lighter traffic becomes higher

The next task was to translate those air volumes to sectors. This needed to be done, due to the fact that capacities can apply to sectors but not to air volumes. This task was complicated by the fact that each volume could correspond to multiple sectors. So, a list was compiled, containing of all sectors (and their capacities) that could possibly be active during the day of interest. To verify which sector was active during the time each air volume was crossed, we cross checked with the active configurations.

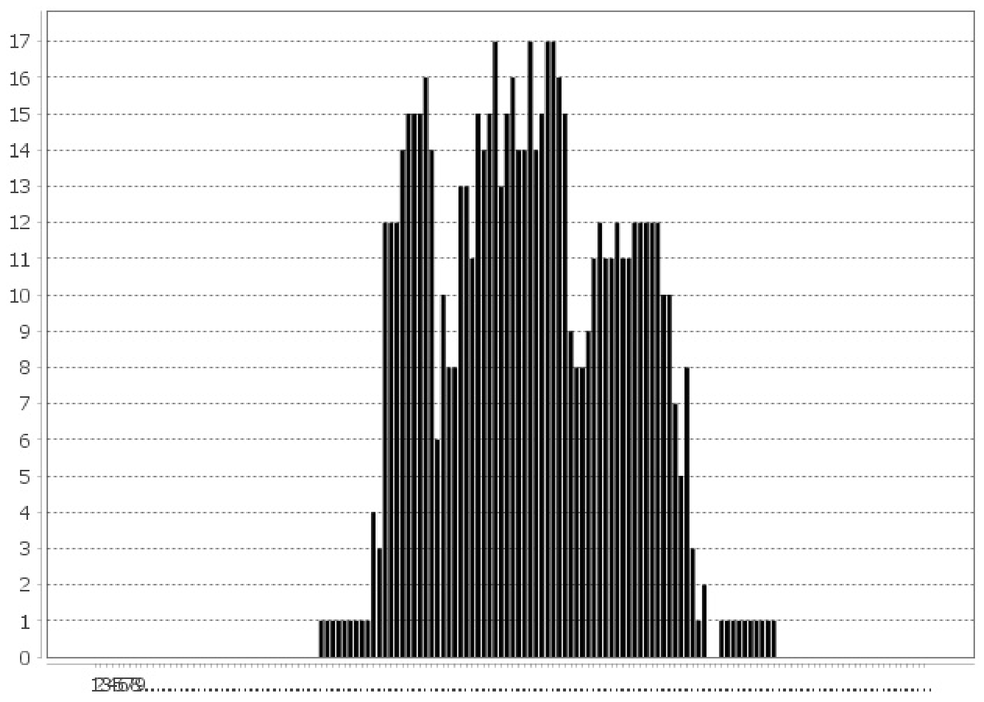


Figure 2.5: Air Traffic during 12/01/2016 in a specific volume. X-axis shows the time and Y-axis the number of flights in the volume

A configuration is a structural element of the air-space right above the sectors. It is active for a period of time, from several minutes to a few hours, and contains a number of sectors. So each volume, given its corresponding time stamps, can belong to only one active sector, which in turn belongs to an active configuration.

After all air volumes were successfully translated to sectors, we had all the vital elements of each flight plan in order to run our experiments.

3. PRELIMINARIES

3.1. Markov Decision Processes

This section provides a general, high-level overview of Markov decision processes. We are concerned with sequential decision problems where there is a need to make many decisions in the lifetime of a system.

We assume that there is a discrete sequence of time points at which we get to make decisions. It is possible to consider continuous time processes, but the issues that arise from this added complexity is not in the scope of this thesis. The problem of calculating a complete mapping from states to actions, which is called a policy, in an accessible, stochastic environment with a known transition model is called a Markov decision problem (MDP), after the Russian statistician Andrei A. Markov. Markov's work is so closely associated with the assumption of accessibility, that decision problems are often divided into ”Markov” and ”non-Markov”. More strictly, we say the Markov property holds if the transition probabilities from any given state depend only on the state and not on previous history.

An MDP is defined by the following three components [11]:

- Initial State: S_0
- Transition Model: $T(s, a, s')$
- Reward unction: $R(s)$

A specification of the outcome probabilities for each action in each possible state is called a transition model. We will use $T(s, a, s')$ to denote the probability of reaching state s' if action a is done in state s. To complete the definition, we must specify the utility function for the agent. For now, we will simply stipulate that in each state s, the agent receives a reward $R(s)$, which may be positive or negative, but must be bounded.

A solution must specify the action that the agent must apply at any state that the agent might reach. A solution of this kind is called a policy, as previously stated. We usually denote a policy by π, and $\pi(s)$ is the action recommended by the policy π for state s. If the agent has a policy, then no matter what the outcome of any action, the agent will always know what to do next.

The quality of a policy is measured by the *expected* utility, calculated by 3.1 (where the utility U of a state s_i is multiplied by the probability P_i of its occurrence [11]) of the possible environment histories generated by that policy.

$$E(U) = \sum_{i=1}^{n} P_i * U(s_i) \tag{3.1}$$

An optimal policy is a policy that yields the highest expected utility. We use π^* to denote an optimal policy. Given π^*, the agent decides what to do by consulting its current precept, which signifies the current state s, and then executing the action $\pi^*(s)$. A policy represents the agent function explicitly and is therefore a description of a simple reflex agent, computed from the information used for a utility-based agent.

The careful balancing of risk and reward is a characteristic of MDPs that does not arise in deterministic search problems; moreover, it is a characteristic of many real-world decision problems. For this reason, MDPs have been studied in several fields, including AI, operations research, economics, and control theory. Dozens of algorithms have been proposed for calculating optimal policies, the most notable being Value Iteration [12] and Policy Iteration [13].

3.2. Reinforcement Learning

This section aims to provide a basic understanding of Reinforcement Learning [11]. We know an agent can learn to play chess by supervised learning, by being given examples of game situations along with the best moves for those situations. But if there is no teacher providing examples, what can the agent do? By trying random moves, the agent can eventually build a predictive model of its environment: what the board will be like after it makes a given move and even how the opponent is likely to reply in a given situation.

The problem is this: without some feedback about what is good and what is bad, the agent will have no grounds for deciding which move to make. The agent needs to know that something good has happened when it wins and that something bad has happened when it loses. This kind of feedback is called a *reward*, or *reinforcement*. In environments like the one we are considering in this thesis, the rewards come frequently. Reinforcement has been carefully studied by animal psychologists for over 60 years.

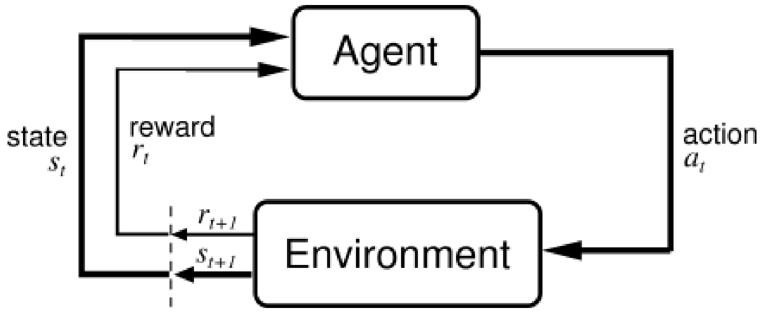


Figure 3.1: The basic premise of Reinforcement Learning

Rewards were introduced in the previous section, where they served to define optimal policies in Markov decision processes (MDPs). An optimal policy is a policy that maximizes the expected total reward. The task of reinforcement learning is to use observed rewards to learn an optimal (or nearly optimal) policy for the agent ot act in the environment.

In many complex domains, reinforcement learning is the only feasible way to train a program to perform at high levels. For example, in game playing, it is very hard for a human to provide accurate and consistent evaluations of large numbers of positions, which would be needed to train an evaluation function directly from examples.

Instead, the program can be told when it has won or lost, and it can use this information to learn an evaluation function that gives reasonably accurate estimates of the probability of winning from any given position. Similarly, it is extremely difficult to program an agent to fly a helicopter; yet given appropriate negative rewards for crashing, wobbling, or deviating from a set course, an agent can learn to fly by itself.

Reinforcement learning might be considered to encompass all of AI: an agent is placed in an environment and must learn to behave successfully therein. There are three main agent designs:

- A *utility-based agent* learns a utility function on states and uses it to select actions that maximize the expected outcome utility. A utility-based agent must also have a model of the environment in order to make decisions, because it must know the states to which its actions will lead. For example, in order to make use of a backgammon utility function, a backgammon program must know what its legal moves are and how they affect the board position. Knowing the actions and their immediate effect on the environment is not enough: the agent has to know where it is headed, and whether it is headed to a winning situation or not. Only in this way can it apply the utility function and choose the optimal action to the outcome states.
- A *Q-learning agent* learns an action-value function, or Q-function, giving the expected utility of taking a given action in a given state. A Q-learning agent can compare the values of its available choices without needing to know their outcomes, so it does not need a model of the environment. On the other hand, because they do not know where their actions lead, Q-learning agents cannot look ahead.
- A *reflex agent* learns a policy that maps directly from states to actions. Simple reflex agents act only on the basis of the current percept, ignoring the rest of the percept history. The agent function is based on the condition-action rule: if condition then action. This agent function only succeeds when the environment is fully observable.

There are also two main divisions of Reinforcement Learning.
Passive Learning, where the agent's policy is fixed and the task is to learn the utilities of states (or state-action pairs); this could also involve learning a model of the environment. The three basic approaches of this division are the following:

- Direct Utility Estimation. A simple method for this was invented in the late 1950s in the area of adaptive control theory by Widrow and Hoff. The idea is that the utility of a state is the expected total reward from that state onward, and each trial provides a sample of this value for each state visited. Thus, at the end of each sequence, the algorithm calculates the observed reward-to-go for each state and updates the estimated utility for that state accordingly, just by keeping a running average for each state in a table.

- An Adaptive Dynamic Programming agent works by learning the transition model of the environment as it goes along and solving the corresponding Markov decision process using a dynamic programming method. For a passive learning agent, this means plugging the learned transition model and the observed rewards into the Bellman equations to calculate the utilities of the states. Because the model usually changes only slightly with each observation, the value iteration process can use the utility estimates calculated up to each point as initial values and should converge quite quickly.
- Temporal Difference Learning combines the best of both previous approaches. The key is to use the observed transitions to adjust the values of the observed states so that they agree with the constraint equations. The basic idea of all temporal-difference methods is, first to define the conditions that hold locally when the utility estimates are correct, and then, to write an update equation that moves the estimates toward this ideal ”equilibrium” equation.

Active Learning, where the agent must also learn what to do. As opposed to a passive learning agent, which has a fixed policy that determines its behavior, the principal issue here is exploration: an agent must experience as much as possible of its environment in order to learn how to behave in it.

An agent of Active Reinforcement Learning must make a trade-off between exploitation to maximize its reward -as reflected in its current utility estimates- and exploration to maximize its long-term well-being. Pure exploitation risks getting stuck in a sub-optimal state. Pure exploration to improve one’s knowledge is of no use if one never puts that knowledge into practice. In the real world, one constantly has to decide between continuing in a comfortable existence and striking out into the unknown in the hopes of discovering a new and better life. With great understanding, less exploration is necessary.

So, an obvious question arises about the existence of an optimal exploration method. This question has been studied in depth in the subfield of statistical decision theory that deals with the well-known bandit problems [14].

One of the simplest approaches is to have the agent choose a random action a fraction 1/t of the time and to follow the greedy policy otherwise. While this does eventually converge to an optimal policy, it can be extremely slow. A more sensible approach would give some weight to actions that the agent has not tried very often, while tending to avoid actions that are believed to be of low utility. Essentially, this amounts to an optimistic prior over the possible environments and causes the agent to behave initially as if there were wonderful rewards scattered all over the place.

3.2.1. Q-Learning

The Active Temporal Difference method is called Q-learning and learns an action-value function instead of learning utilities. We will use the notation $Q(a, s)$ to denote the value of doing action a in state s. Q-values are directly related to utility values as shown in equation 3.2:

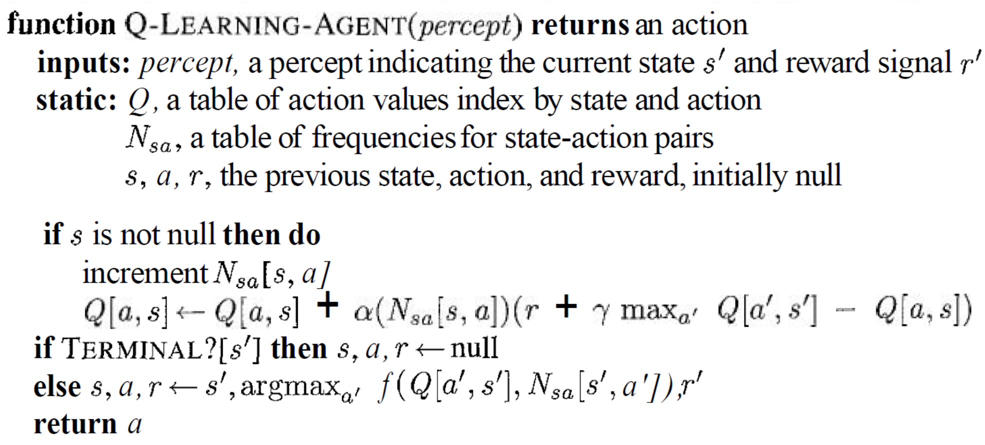
$$U(s) = \max_a Q(a, s) \tag{3.2}$$

Q-functions may seem like just another way of storing utility information, but they have a very important property: a Temporal Difference agent that learns a Q-function does not need a model for either learning or action selection. For this reason, Q-learning is called a model-free method. The update equation for Temporal Difference Q-learning, which is calculated whenever action a is executed in state s leading to state s', is 3.3:

$$Q(a, s) := Q(a, s) + \alpha[Rwd(s) + \gamma \max_{a'} Q(a', s') - Q(a, s)] \tag{3.3}$$

Here α is called learning rate and determines to what extent the newly acquired information will override the old information. A factor of 0 will make the agent not learn anything, while a factor of 1 would make the agent consider only the most recent information. The discount factor γ determines the importance of future rewards. A factor of 0 will make the agent short-sighted by only considering current rewards, while a factor approaching 1 will make it strive for a long-term high reward. If the discount factor meets or exceeds 1, the action values may diverge.

Since Q-learning is an iterative algorithm, it implicitly assumes an initial condition before the first update occurs. High initial values, also known as ”optimistic initial conditions”, can encourage exploration: no matter what action is selected, the update rule will cause it to have lower values than the other alternative, thus increasing their choice probability. Figure 3.2 presents the behavior of a basic Q-Learning agent in pseudo code.



```
function Q-LEARNING-AGENT(percept) returns an action
   inputs: percept, a percept indicating the current state $s'$ and reward signal $r'$
   static: $Q$, a table of action values index by state and action
           $N_{sa}$, a table of frequencies for state-action pairs
           $s$, $a$, $r$, the previous state, action, and reward, initially null

   if $s$ is not null then do
      increment $N_{sa}[s, a]$
      $Q[a, s] \leftarrow Q[a, s] + \alpha(N_{sa}[s, a])(r + \gamma \max_{a'} Q[a', s'] - Q[a, s])$
   if TERMINAL?[$s'$] then $s, a, r \leftarrow$ null
   else $s, a, r \leftarrow s', \text{argmax}_{a'} f(Q[a', s'], N_{sa}[s', a']), r'$
   return $a$
```

Figure 3.2: An exploratory Q-learning agent. It is an active learner that learns the value $Q(a, s)$ of each action in each situation.

4. 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 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 4.1, 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*$)\}$

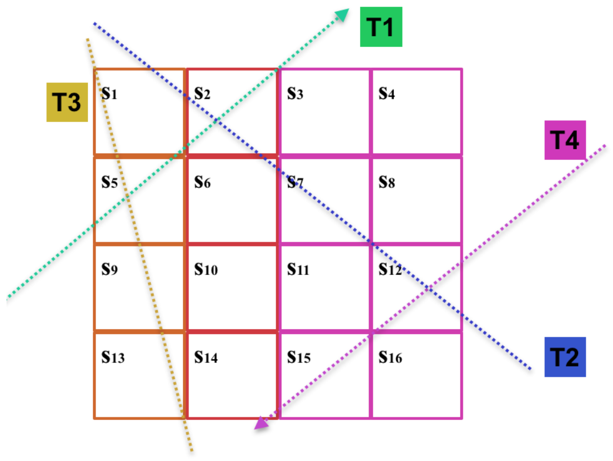


Figure 4.1: Example of trajectories crossing sectors

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 4.1, 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 $\gamma(i) : D_i \to \mathbb{R}$. 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 thesis 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:

- 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

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

4.1. The MDP Framework

In principle, a collaborative multiagent MDP can be regarded as one large single agent in which each joint action is represented as a single action. It is then possible to learn the optimal $Q-$values for the joint actions using standard single-agent $Q-$learning. In this MDP, either a central controller models the complete MDP and communicates to each agent its individual action, or each agent models the complete MDP separately and selects the individual action that corresponds to its own identity.

In the latter case, the agents do not need to communicate but they have to be able to observe the executed joint action and the received individual rewards. The problem of exploration is solved by using the ϵ-greedy exploration-exploitation strategy for all agents [15].

Although this approach leads to the optimal solution, it is infeasible for problems with many agents. In the first place, it is intractable to model the complete joint action space, which is exponential in the number of agents. For example, a problem with 20 agents, here flights, each able to perform 2 actions (add a delay or not) results in more than one million $Q-$values per state. Secondly, the agents might not have access to the needed information for the update because they are not able to observe the state, action, and reward of all other agents. Finally, it will take many time steps to explore all joint actions resulting in slow convergence.

So, in order to exploit its various advantages, we use the model of collaborative multiagent MDP framework [16], [17] which assumes:

-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 generalized 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^t_A (e.g. $str^t_{N(A_i)}$). 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 $p(s^{t+1}_i | s^t_i, str^t_i)$.

-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^t_A in their joint state s^t_A, i.e. according to their decided delays, and also according to their preferences on the chosen delays. Formally:

$$Rwd_A(s^t_A, str^t_A) = \lambda_1 * X(str^t_A, s^t_A) + \lambda_2 * D(str^t_A, s^t_A) \tag{4.1}$$

where, $X(str^t_A, s^t_A)$ 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), $D(str^t_A, s^t_A) : s_A \rightarrow \mathbb{R}$, 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 π_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

$$V_i^*(s) = max_{\pi_i} E[\sum_{t=0}^{\infty} \delta^t Rwd_i(s_i^t, \pi_i(s_i^t))|\pi_i)] \tag{4.2}$$

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.

5. COLLABORATIVE REINFORCEMENT LEARNING ALGORITHMS

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.

5.1. Independent Reinforcement Learners

The independent learners Q-learning variant proposed in [18] 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

$$Q_i(s_i, str_i) := Q_i(s_i, str_i) + \alpha[Rwd(s_{N(A_i)}, str_{N(A_i)}) + \delta max'_a Q(s'_i, str^*_i) - Q(s_i, str_i)] \quad (5.1)$$

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 [18], we use the reward received by the agent, taking into account only the joint state and joint strategy of its neighbourhood.

5.2. Sparse Cooperative Q-Learning

The next two algorithms that we propose in this thesis are based on Sparse Cooperative $Q-$learning, or SparseQ, method which also approximates the global $Q-$function into a linear combination of local $Q-$functions. The decomposition is based on the structure of a Coordinated Graph(CG) which is chosen beforehand.

Here, in order to construct this CG, we assume the following:

- Each Flight is a node of the graph.

- An edge between two graph nodes exists if and only if the two corresponding flights are involved in the same hotspot.

Two connected flights on that graph are called neighbors. A flight can be a neighbor of another flight only if they both can be found at the same sector, at the same occupancy period. It has to be noted here, that given a maximum delay for each flight, the set of possible interacting flights at a time slot is finite.

We investigate both a decomposition in terms of the nodes (or agents), as well as the edges. In the agent-based decomposition the local function of an agent is based on its own action and those of its neighboring agents. In the edge-based decomposition each local function is based on the actions of the two neighbor agents it is connected to. In order to update a local function, the key idea is to base the update not on the difference between the current global $Q-$value and the experienced global discounted return, but rather on the current local $Q-$value and the local contribution of this agent to the global return. In this thesis we utilize the edge-based decomposition.

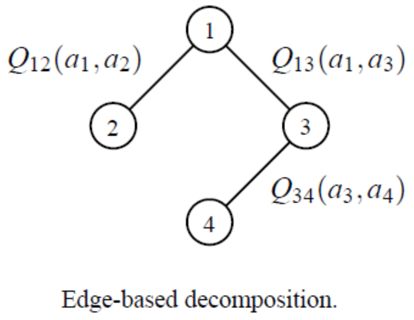


Figure 5.1: An edge-based decomposition of the global Q-function for a 4-agent problem.

5.2.1. Edge-Based Collaborative Reinforcement Learners

This is a variant of the edge-based update sparse cooperative edge-based Q-learning method proposed in [1]. 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:

$$Q_{i,j}(s_{i,j}, str_{i,j}) = Q_{i,j}(s_{i,j}, str_{i,j}) + \alpha(\frac{Rwd_i(s_i, str_i)}{N(A_i)} + \frac{Rwd_j(s_j, str_j)}{N(A_j)} + \delta Q_{i,j}(s'_{i,j}, str^*_{i,j}) - Q_{i,j}(s_{i,j}, str_{i,j})) \quad (5.2)$$

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* [19].

This algorithm is a popular method for computing the maximum a posteriori configuration in an undirected graphical model. It operates by iteratively sending locally optimized messages $_ij(a_j)$ between node i and j over the corresponding edge in the graph. After convergence, each node then computes the MAP assignment based on its local incoming messages only [1].

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

$$\mu_{ij}(str_j) = max_{str_i}\{Q_i(str_i) + Q_{ij}(str_i, str_j) + \sum_{k \in N(A_i)-A_j} \mu_{ki}(str_i)\}, \quad (5.3)$$

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

$$str_i^* = argmax_{str_i}(f_i(str_i) + \sum_{j \in N(A_i)} \mu_{ji}(str_i)) \quad (5.4)$$

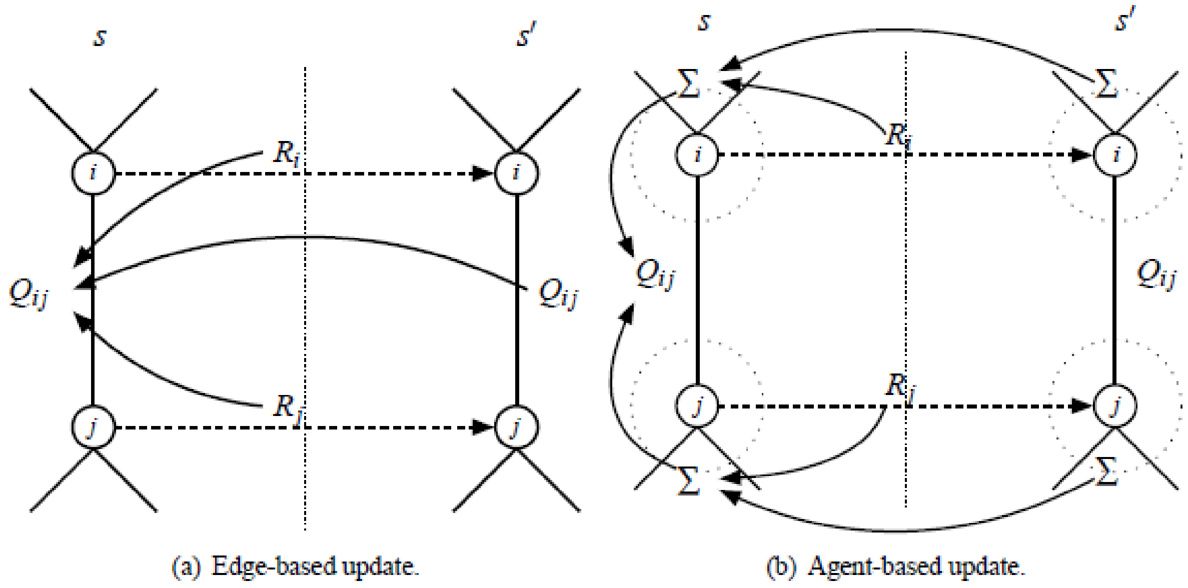


Figure 5.2: A graphical representation of the edge-based and agent-based update method after the transition from state s to s′ [1].

5.2.2. Agent-Based Collaborative Reinforcement Learners

This is a variant of the agent-based update sparse cooperative edge-based Q-learning method proposed in [1]. 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:

$$Q_{i,j}(s_{i,j}, str_{i,j}) = Q_{i,j}(s_{i,j}, str_{i,j}) + \alpha \sum_{k\in\{i,j\}} \frac{(Rwd_{i,j}(s_{i,j}, str_{i,j}) + \delta Q_k(s'_k, str^*_k) - Q_k(s_k, str_k))}{|N(A_k)|} \quad (5.5)$$

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.

6. EXPERIMENTAL EVALUATION

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 6.1 presents the data used in producing the experimental cases and the parameter values used in all simulated runs.

Table 6.1: 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 0.9, 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 6.1 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.

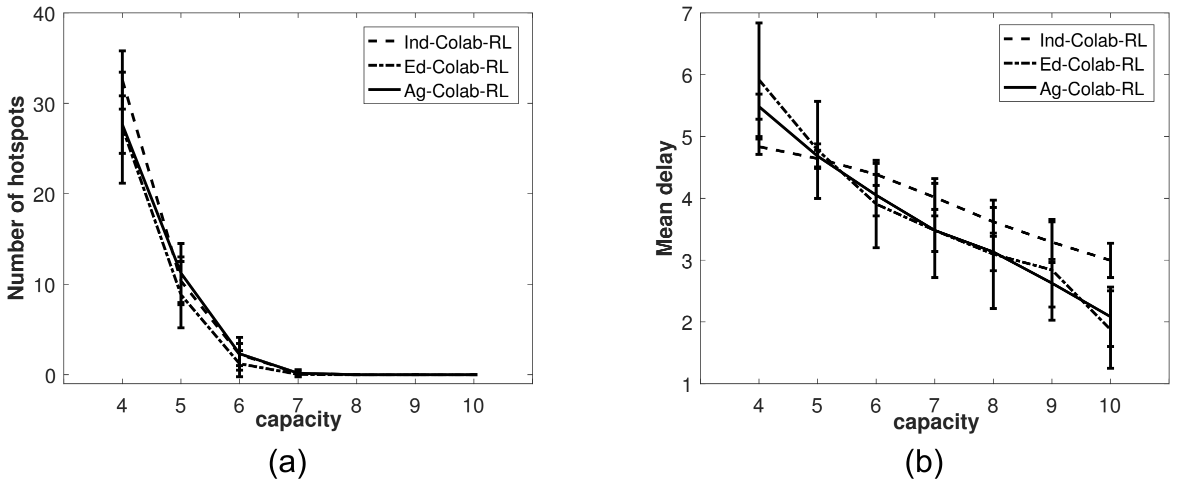


Figure 6.1: Comparative results: Plots illustrate (a) the number of hotspots and (b) the mean delay estimated by each method in terms of various values of sectors' capacity (x-axis).

According to the results and as shown in Fig. 6.1 (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. 6.1 (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. 6.1 (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.

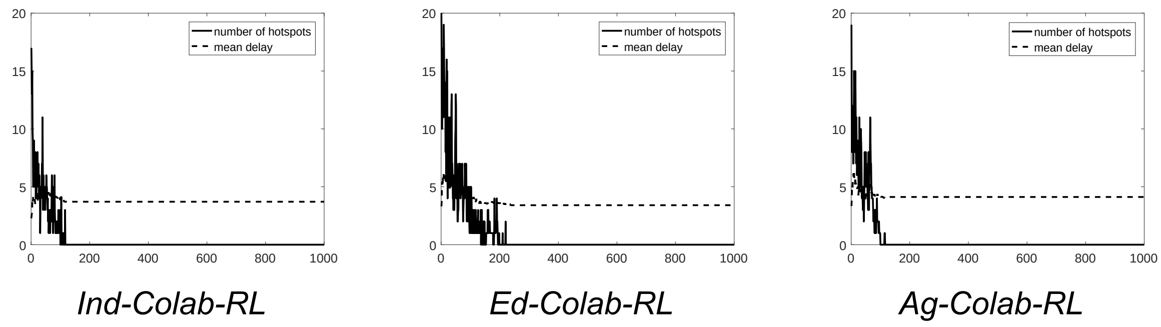


Figure 6.2: Learning curves received by three methods in a setting considering sectors' capacity equal to 7. The x-axis shows the number of the learning episode, while the y-axis shows the number of hotspots and mean delay achieved in each episode.

Figure 6.2 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 exploitation 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).

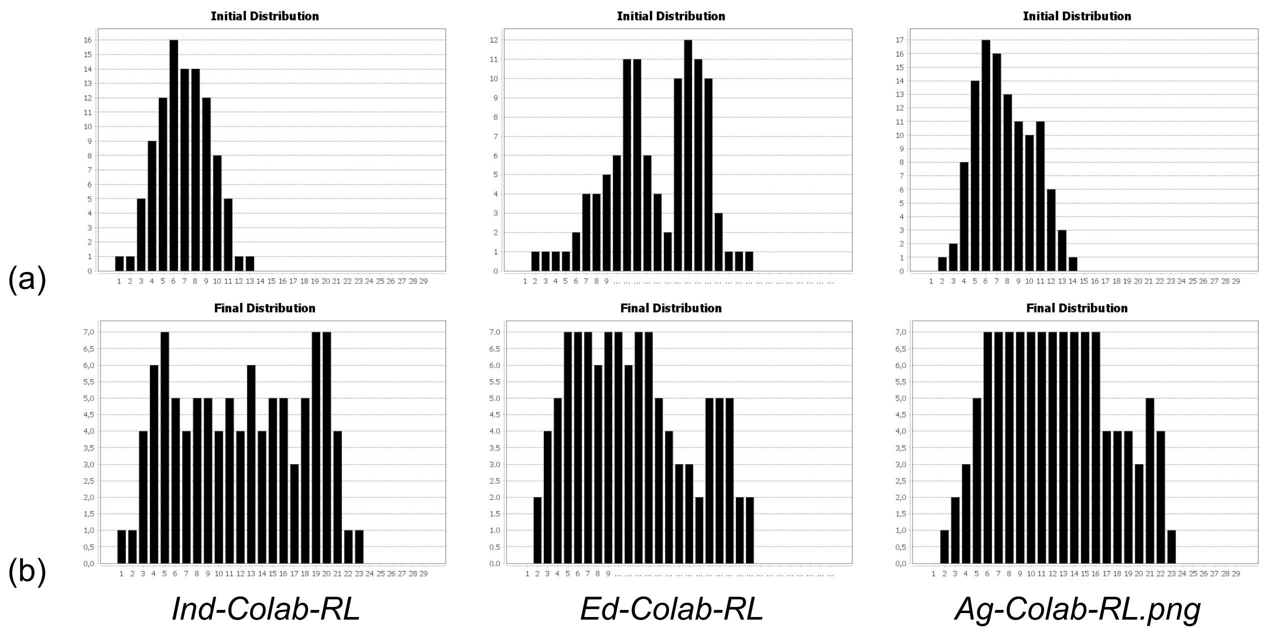


Figure 6.3: An example of the distribution of interacting flights in Occupancy Counting Periods (a) initially and (b) as produced by three methods

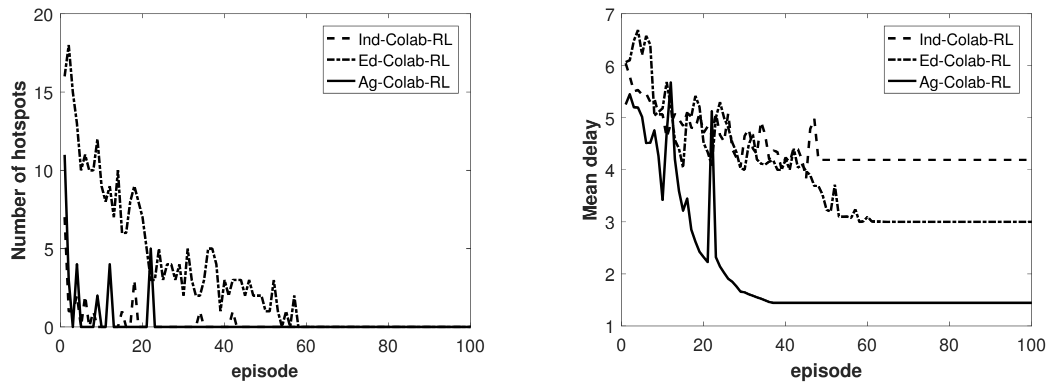
Finally, in Figure 6.3 we present an example of the distribution of hotspots (y-axis) in terms of Occupancy Counting Periods in a number of 29 non-overlapping occupancy periods, each of duration equal to 6 time instants (e.g. 6 minutes). This was obtained by measuring the interacting flights to a specific 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 6.4 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 6.2.

Table 6.2: Parameter values used during the simulated experiments with 3000 flights

Parameter	value
grid structure of sectors	4×4
capacity of sectors, C	20
number of planes, N	3000
Duration and Step of Occupancy Counting Period	6
total time period duration H	180
maximum delay	10

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 6.4(a)) and the mean delay achieved per method (Figure 6.4(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*

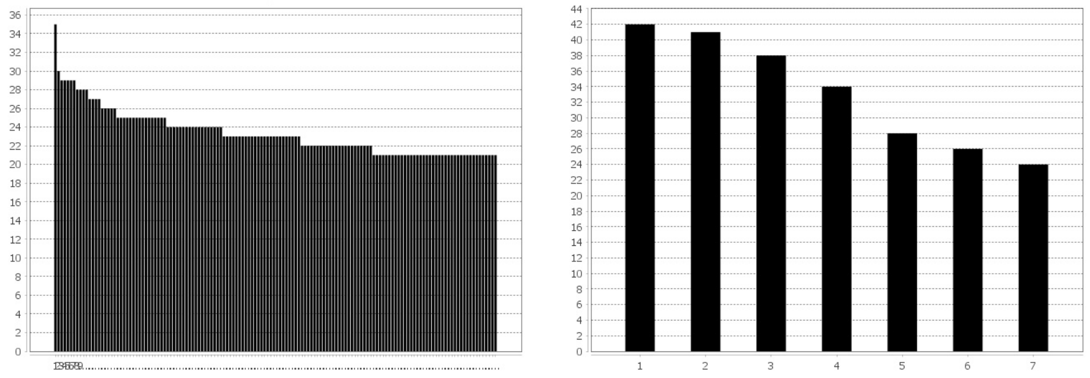
Figure 6.4: Learning curves received by the three methods in a setting where N=3000 and sectors' capacity C=20. The x-axis shows the number of the learning episode, while the y-axis shows (a) the number of hotspots and (b) the mean delay achieved in each episode.

The final experiment was created by utilizing the real world data provided by CRIDA. Table 6.3 specifies the parameters of the scenario that was formulated from the data of the 12th of January 2016. The main difference here, regarding the parameters, is that the delays applied are no longer a multiple of the occupancy period, but plain minutes.

Table 6.3: Parameter values used during the experiments on the real scenario

Parameter	value
grid structure of sectors	11×11
capacity of sectors, C	$\in [15, 53]$
number of planes, N	3195
Duration and Step of Occupancy Counting Period	60
total time period duration H	1440
maximum delay	45

This change brings the experiment closer to a real world situation, but poses an advanced difficulty for two reasons. Firstly, the maximum delay is a much bigger number, which means that every agent has many more states to explore. Secondly, a flight can be delayed for less than one occupancy period, as opposed to the previous experiments, where we could delay a flight for as many as ten periods.



(a) Hotspots in Simulation *(b) Hotspots in Real Scenario*

Figure 6.5: A comparison of the number of hotspots. (a) shows all the hotspots for the 3000 flight simulated scenario and (b) those of the real scenario

On the other hand, at Figure 6.5 we see the hotspots that appear in the simulated 3000 flight scenario, as well as those from the actual one. It is quite obvious that the real problem is much easier than the simulated one, due to the fact that the simulation has tens of hotspots, but the real scenario has only 7. This is a tremendous difference, which shows that during our simulations we pushed our methods to the limit, by solving problems much more difficult than the one produced by the dataset.

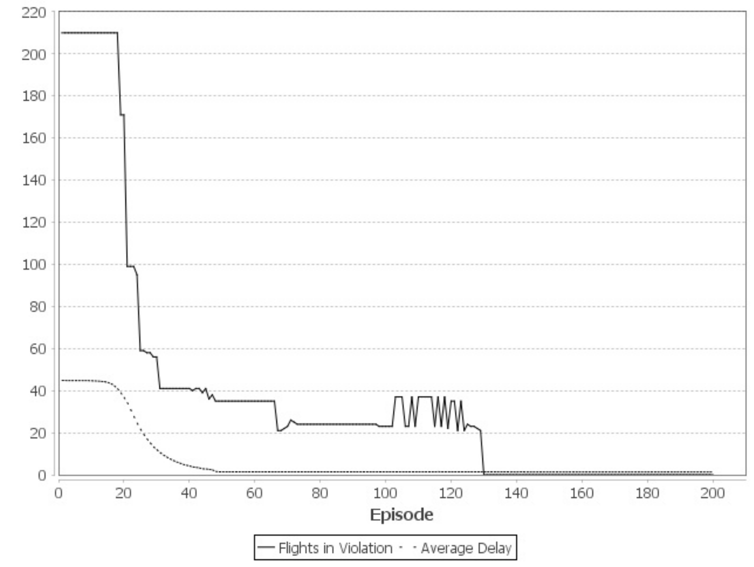


Figure 6.6: Learning curve received by the Independent Learners method in the scenario produced by the dataset. The x-axis shows the number of the learning episode, while the y-axis shows the number of flights in hotspots and mean delay achieved in each episode.

Figure 6.6 shows the learning curve received by the Independent Learners method, which converges to a solution where the average delay is close to 0. The exploration-exploitation policy used here was the ϵ-greedy strategy. The exploration stops at episode 130, where the pure exploitation begins. Figure 6.7 shows the initial and final distribution of flights in the sector with two out of seven total hotspots.

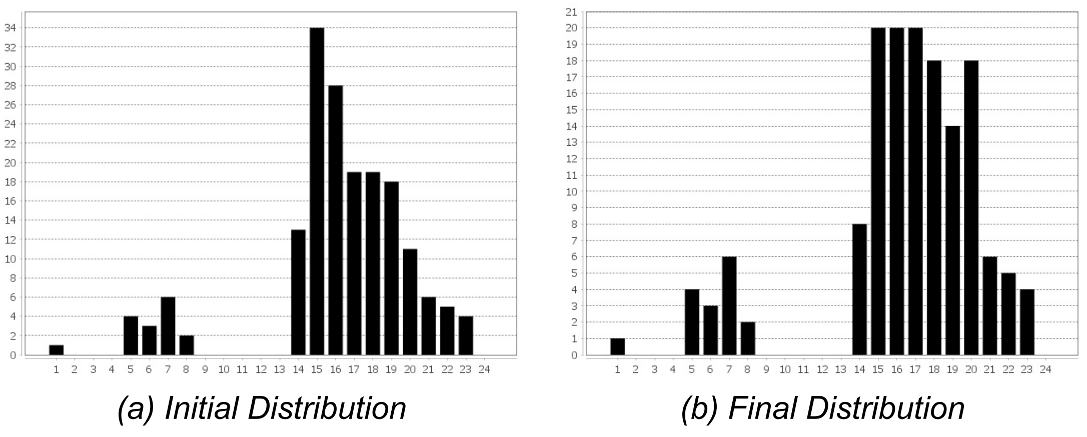


Figure 6.7: An example of the distribution of interacting flights in Occupancy Counting Periods (a) initially and (b) as produced by the Independent Learners. Note that the sector's capacity is 20

7. 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 [20] or geometric approaches [21].

For instance, following an agent-based approach, [22] and [23] 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 [24] 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, [25] 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 [26]. Like [1], the approach employs a variant of max-plus [19] for message-passing, but, crucially, it is able to extend single-agent and centralized multi-agent Bayesian RL methods [27] in collaborative settings by decomposing the coordination problem into *regional* sub-problems.

8. CONCLUSIONS AND FUTURE WORK

In this work we investigated a collaborative reinforcement learning framework for pre-tactical planning of flights, with the aim of eliminating hotspots by applying 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 aircraft is treated as an agent. Three multiagent RL schemes were studied, the Independent Learners, the Edge-Based Collaborative Reinforcement Learners and the Agent-Based Collaborative Reinforcement Learners. As shown in the experimental evaluation section, their preliminary results were quite promising. This was supported further by studying a real world example.

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:

- Preparing more 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.
- Exploiting more historical data to train our methods and compute solutions using them.
- Tune/learn a reward model.
- Compare delays imposed by our methods to those imposed by domain experts in real-life scenarios.
- Utilize Inverse Reinforcement Learning techniques to verify our reward model.

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. Moreover, it is possible to attempt to solve this problem by utilizing algorithms from the Population Based family, like Genetic or Swarm Based algorithms [28].

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

Table 8.1: LIST OF ABBREVIATIONS

AI	Artificial Intelligence
RL	Reinforcement Learning
ATM	Air Traffic Management
TCB	Trajectory-Based Operations
ANS	Air Navigation System
ASM	Air Service Management
ATC	Air Traffic Control
DCB	Demand Capacity Balance
MDP	Markov Decision Process
Ind-Colab-RL	Independent Collaborative Reinforcement Learners
Ed-Colab-RL	Edge-Based Collaborative Reinforcement Learners
Ag-Colab-RL	Agent-Based Collaborative Reinforcement Learners

REFERENCES

[1] Jelle R. Kok and Nikos Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *J. Mach. Learn. Res.*, 7:1789–1828, December 2006.

[2] Theocharis Kravaris, George A. Vouros, Christos Spatharis, Konstantinos Blekas, Georgios Chalkiadakis, and Jose Manuel Cordero Garcia. Learning policies for resolving demand-capacity imbalances during pre-tactical air trafficc management. 2017.

[3] ICAO, DOC. 9613 AN/937. Performance--based navigation manual. Vol. I. Concept and implementation guidance, Vol. II. Implementing RNAV and RNP operations, 2013.

[4] Manuel Soler. Fundamentals of Aerospace Engineering: An introductory course to aeronautical engineering, 2014.

[5] ENAIRE, 2016. `www.enaire.es`.

[6] Annex, ICAO. Meteorological Service for International Air Navigation, International Civil, 2010.

[7] L. P. Sanz, R. M. A. Valdés, F. J. S. Nierto, J. B. Monge y V. F. G. Comendador. Introducción al Sistema de Navegación Aérea.

[8] E. R. Mueller y G. B. Chatterji. Analysis of aircraft arrival and departure delay characteristics, 2002. from AIAA aircraft technology, integration and operations (ATIO) conference.

[9] University of Westminster, Eurocontrol. European airline delay cost reference values, 2010. p. 86.

[10] EUROCONTROL. Air traffic flow and capacity management (atfcm), 2011. `www.eurocontrol.int/articles/air-traffic-flow-and-capacity-management`.

[11] Peter Nowig Stuart Russell. *Artificial Intelligence A Modern Approach Second Edition*. Pearson Education Inc, Upper Saddle River, New Jersey , USA, 2003.

[12] Elena Pashenkova and Irina Rish and Rina Dechter. Value iteration and policy iteration algorithms for Markov decision problem, 1996.

[13] Michail G. Lagoudakis and Ronald Parr. Least-Squares Policy Iteration, 2003.

[14] Dirk Bergemann and Juuso Välimäki. Bandit Problems, 2006.

[15] N. Vlassis. A concise introduction to multiagent systems and distributed ai, 2003.

[16] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[17] Carlos Ernesto Guestrin. *Planning Under Uncertainty in Complex Structured Environments*. PhD thesis, Stanford, CA, USA, 2003. AAI3104233.

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

[19] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[20] Kwangyul Baek and Hyochoong Bang. Ads-b based trajectory prediction and conflict detection for air traffic management. *International Journal Aeronautical and Space Sciences*, 13(3):377–385, 2012.

[21] Martina Orefice, Vittorio Di Vito, Federico Corraro, Giancarmine Fasano, and Domenico Accardo. Aircraft conflict detection based on ads-b surveillance data. In *Metrology for Aerospace (MetroAeroSpace), 2014 IEEE*, pages 277–282. IEEE, 2014.

[22] Baraa Munqith Albaker and Nasrudin Abd Rahim. Unmanned aircraft collision avoidance system using cooperative agent-based negotiation approach. *Int. J. Simulation, Syst. Sci. Technol*, 11(4):1–8, 2010.

[23] David Sislak, Přemysl Volf, and Michal Pechoucek. Agent-based cooperative decentralized airplane-collision avoidance. *IEEE Transactions on Intelligent Transportation Systems*, 12(1):36–46, 2011.

[24] Shawn R Wolfe, Peter A Jarvis, Francis Y Enomoto, Maarten Sierhuis, and Bart-Jan van Putten. A multi-agent simulation of collaborative air traffic flow management. In *Multi-Agent Systems for Traffic and Transportation Engineering*, pages 357–381. IGI Global, 2009.

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

[26] W. T. L. Teacy, G. Chalkiadakis, A. Farinelli, A. Rogers, N. R. Jennings, S. McClean, and G. Parr. Decentralized bayesian reinforcement learning for online agent collaboration. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, pages 417–424, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[27] G. Chalkiadakis and C. Boutilier. Coordination in Multiagent Reinforcement Learning: A Bayesian Approach. In *Proc. of AAMAS 2003*, pages 709–716, 2003.

[28] Ilhem Boussaïd and Julien Lepagnot and Patrick Siarry. A survey on optimization metaheuristics, 2013.