Neural Computing and Applications (2023) 35:147–159
https://doi.org/10.1007/s00521-021-05748-7

S.I. : INFORMATION, INTELLIGENCE, SYSTEMS AND APPLICATIONS

Hierarchical multiagent reinforcement learning schemes for air traffic management

Christos Spatharis[1] · Alevizos Bastas[2] · Theocharis Kravaris[2] · Konstantinos Blekas[1] · George A. Vouros[2] · Jose Manuel Cordero[3]

Received: 30 January 2020 / Accepted: 16 January 2021 / Published online: 10 February 2021

Abstract

In this work we investigate the use of hierarchical multiagent reinforcement learning methods for the computation of policies to resolve congestion problems in the air traffic management domain. To address cases where the demand of airspace use exceeds capacity, we consider agents representing flights, who need to decide on ground delays at the pre-tactical stage of operations, towards executing their trajectories while adhering to airspace capacity constraints. Hierarchical reinforcement learning manages to handle real-world problems with high complexity, by partitioning the task into hierarchies of states and/or actions. This provides an efficient way of exploring the state–action space and constructing an advantageous decision-making mechanism. We first establish a general framework of hierarchical multiagent reinforcement learning, and then, we further formulate four alternative schemes of abstractions, on states, actions, or both. To quantitatively assess the quality of solutions of the proposed approaches and show the potential of the hierarchical methods in resolving the demand–capacity balance problem, we provide experimental results on real-world evaluation cases, where we measure the average delay per flight and the number of flights with delays.

Keywords Multiagent reinforcement learning · Hierarchical learning · State abstraction · Congestion problems · Air traffic management

1 Introduction

Congestion problems deal with situations where resources of a limited capacity have to be shared simultaneously by multiple agents. They can be found in a wide range of domains in the modern world, plaguing various aspects of our business, activities, and daily lives. Air traffic management (ATM) is an important domain where congestion problems appear, implying costs and uncertainty to the scheduling of operations. More specifically, congestion problems arise naturally whenever demand of airspace use exceeds capacity, resulting in hotspots. This is known as the *demand–capacity balance (DCB)* problem or process.[1] The current ATM system worldwide is based on a time-based operations paradigm that leads to DCB issues. These are resolved via airspace management or flow management solutions, including regulations that generate delays. These cascade to the entire system, increasing uncertainty of operations and thus costs. Nowadays, demand–capacity

✉ George A. Vouros
georgev@unipi.gr

Christos Spatharis
cspatharis@cse.uoi.gr

Alevizos Bastas
alevizos.b@gmail.com

Theocharis Kravaris
hariskrav.unipi@gmail.com

Konstantinos Blekas
kblekas@cse.uoi.gr

Jose Manuel Cordero
jmcordero@e-crida.enaire.es

[1] Department of Computer Science and Engineering, University of Ioannina, Ioannina, Greece

[2] Department of Digital Systems, University of Piraeus, Piraeus, Greece

[3] CRIDA, Madrid, Spain

[1] http://www.eurocontrol.int/articles/air-traffic-flow-and-capacity-management.

imbalances are difficult to be predicted at the pre-tactical phase (prior to operation) as the existing information regarding operations is not accurate enough during this phase.

Congestion problems have been extensively studied in game theoretic models [29, 30, 34, 38], in the optimisation, transportation research, automatic control literature [17], as well as in the AI and autonomous agents research for at least two decades now [2, 7, 21, 26]. Multiagent reinforcement learning (MARL) has proved to be a suitable framework for such problems [9, 12, 26, 35, 44] as it allows autonomous agents to learn in a decentralised manner congestion resolution policies by interacting with a common environment.

In our previous articles [20, 40, 41] we have formulated the task of resolving demand–capacity imbalances in ATM as a coordination problem of agents that operate to a multiagent Markov decision process (MDP). The interacting agents, representing flights, aim to decide on own ground delays, jointly with others, based on their own preferences and operational constraints on the use of airspace. Their main goal is to reach an equilibrium to conflicting delay preferences, while resolving hotspots in which they participate. We consider the "pre-tactical" stage of air traffic management (ATM) operations, where resources correspond to air sectors, and operational constraints concern sectors' limited capacity. We are mostly interested in minimising scheduled flight delays and thus delay costs.

In this study we propose a general hierarchical multiagent reinforcement learning framework and corresponding hierarchical schemes, which have not been examined in any of the previous works. These allow the introduction of abstraction schemes in state and/or action spaces, the construction of multiple policies at different levels of abstraction and the capabilities to explore efficiently the original (ground) space towards reaching high-quality solutions.

Given the large number of flights per day above Europe, and the number of options per flight to resolve DCB problems, the state–action space increases exponentially to the number of flights and their delay options. Abstraction (or aggregation) is a well-known approach in the field of artificial intelligence to effectively reduce the state/action space, increasing computational efficiency and supporting the computation of qualitative solutions. Instead of working in the ground state space, the decision-maker usually finds solutions in the abstract state space much faster, by treating groups of states as a unit, ignoring irrelevant state information. The same process can be also applied in the action space, where action abstraction (e.g. via *temporal abstraction*) can increase the effective search depth by considering high-level actions composed from many concrete actions [32, 42]. Thus, abstraction can appear simultaneously in the state and action spaces, or in only one of them.

Authors in [1, 23] provide a unified treatment of state abstraction for Markov decision processes and study five particular abstraction schemes. Based on these schemes different hierarchical approaches can be distinguished depending on the aggregation function used (e.g. reward or model based, Q-value based, etc.). In this work we employ abstractions that exploit states' temporal dimension, which is in contrast to other approaches that use temporal abstractions (e.g. [13, 42]) but incorporate temporally extended subtasks in the set of available actions. We aggregate states' temporal dimension in a direct way, e.g. by treating temporal instants as an aggregated temporal "point". With the term "hierarchical" we refer to the hierarchy of such abstractions. Thus, our work is closer to the state abstraction methods, grouping together states with similar configurations and associated behaviour. However, and in conjunction to state abstractions, we do study abstractions in the action space, as well.

Generally, abstraction methods may require human supervision (e.g. [22, 33]), or they may support discovering appropriate abstractions (e.g. as in [16, 19, 27, 28]), automatically. These objectives are out of present work's scope.

It must be noted that this article substantially improves and extends our recent work presented in [39]. Its contributions can be summarised as follows:

- The demand–capacity balance problem is formulated as a hierarchical multiagent Markov decision process at multiple levels of abstraction.
- A generic multiagent hierarchical reinforcement learning framework is devised, able to operate at multiple levels of abstraction.
- The hierarchical framework proposed is instantiated to alternative hierarchical schemes, exploiting multiple levels of abstraction, both at the action and the state space, regarding the DCB problem. To our knowledge, this is the first study that systematically proposes several hierarchical multiagent reinforcement learning schemes to resolve real-world congestion problems, applied in the context of the complex DCB process in the ATM domain.
- All hierarchical multiagent reinforcement learning methods are evaluated in real-world cases comprising large number of flights. The data sources used to produce those cases include real-world operational data regarding flight plans per day of operation, data regarding airspace configurations at any given time, and reference values for the cost of strategic delay to

European airlines, currently used by SESAR 2020 Industrial Research [10].

The remainder of this article is structured as follows: Section 2 offers a brief review of previous works. In Sect. 3 we introduce a specification for the generic congestion problem and present the ATM demand–capacity problem. Section 4 initially presents a generic framework for hierarchical multiagent reinforcement learning methods with multiple abstraction levels and then four specific methods. Section 5 presents evaluation cases and experimental results, and finally, Sect. 6 concludes the article and outlines future research directions.

2 Related work

Hierarchical reinforcement learning constitutes a long-standing study topic. In the literature there are some important initial works, such as the hierarchies of abstract machines (HAM) [32], the "option" framework [42], the feudal networks [11] and the MAX-Q-learning method [13]. More specifically, in [42] the authors introduce the options framework, where temporal abstractions over the action space are exploited, extending the MDP framework to a semi-Markov decision process (SMDP). The agent is responsible for selecting either a primitive or a multi-step action, called option, and each option is described by a policy over the actions and a stochastic termination function. In a recent extension of the "options" framework [5], a policy gradient approach, named "option-critic", is proposed, where the agent learns the "options" by itself. Furthermore, in [37] the authors extended the previous option-critic framework and presented the first general purpose reinforcement learning architecture to successfully learn options from data with more than two abstraction levels. In [11] a set of high-level managers are accountable for setting tasks to sub-managers, who in turn, learn how to satisfy those tasks. A hierarchical reinforcement learning decomposition (value function decomposition into combination of value functions) method based on the MAX-Q algorithm is presented in [13], and in [25] the authors extend the previous framework into multiagent problems. Authors in [22] propose the use of intrinsic behaviours and suggest a meta-controller for deciding the sub-goals and the use of low-level controllers—in the form of neural networks—for choosing appropriate actions for each sub-goal. Later, in [6] the authors took advantage of the state abstraction by extending the "options" framework to Partially Observable MDPs (POMDPs) and they developed a hierarchical Monte Carlo Tree Search (MCTS) algorithm for approximately solving the abstract POMDP. Recently, in [24] the authors created several environments of unmanned vehicles swarms with multiple objectives and they introduced a hierarchical reinforcement learning algorithm, named Dynamic Domain Reduction for Multi-agent Planning, that simultaneously searches in sub-environments and yields sequences of actions with the greatest expected reward. Finally, in [8] a deep hierarchical reinforcement learning algorithm for temporal delayed problems is proposed. Specifically, using a hierarchical policy gradient method, authors train an autonomous driving agent in a traffic light passing scenario, where the agent had two distinctive behaviours (pass and stop) and several primitive actions (acceleration options).

Learning and operating over different levels of temporal abstraction constitutes a challenge in several tasks. Inspired from the work of Sutton et al. [42], in [3] a policy sketches scheme is defined, which annotates tasks with sequences of subtasks, and learns the subtasks and upper-level tasks, jointly. Moreover, recently there are some works that seek to learn the temporal abstraction with deep learning [14, 31, 43].

Our study here aims to bridge the approaches described above, focusing on providing a general framework for hierarchical multiagent reinforcement learning. Although it is employed in the ATM domain, it remains general enough over multiple objective domains. Our primary goal is (a) to construct a generic multiagent framework that supports abstractions at several levels of the state and the action spaces, or any of them, using different abstraction methods, (b) while enabling agents to apply either a coordinated learning process, or to learn independently from the others (i.e. treating the others as part of their environment). We configured alternative methods to achieve these goals, introducing different hierarchical multiagent reinforcement learning schemes, in order to experimentally study their potential to provide qualitative solutions to important congestion problems in ATM.

3 Demand–capacity balance problem in ATM

Today the Air Traffic Management (ATM) system shows demand–capacity imbalances resolved via solutions that mainly include regulations, and generate delays and costs for the entire system. The DCB process is divided in three phases: Strategic, Planning and Tactical Phase. The objective is to design optimum traffic flows based on air traffic control capacity, while enabling airlines to operate safe and efficient flights. In this work we consider DCB at the pre-tactical stage, considering flight plans, i.e. intended flights' trajectories.

The DCB problem considers two important aspects of the ATM system: the *aircraft trajectories* and the *airspace*

sectors. *Sectors* constitute a significant element that can be considered as air volumes segregating the airspace and can be defined as group of airlocks. Sequentially, airblocks are specified by their geometry (i.e. perimeter of their projection on earth), and their lowest and highest altitudes. Airspace sectorisation may be represented in alternative ways that depend on sector configuration and the number of active (open) sectors. Only one sector configuration can be active at a time. However, during a single day the sectorisation of airspace can change frequently according to different operational conditions and needs. A more comprehensive description on these issues can be found in [40, 41].

A significant quantity in our study is the *capacity of sectors* that determines the maximum number of flights passing from a sector during a specific time period. The *demand* for each sector specifies the number of flights that co-occur during any time period within a sector. Demand must not exceed sector capacity for any time period. Several types of measures can be used to monitor the demand evolution. In this work we consider the *Entry Count* that gives the number of flights entering the sector during a time period. Besides, this is used by network manager (NM) at the pre-tactical stage.

The *counting period* for a given sector is defined as the number of flights entering the sector during a time period. It gives a "picture" of the entry traffic, taken at every time "step" value along a period of fixed duration. The *counting step* defines the time difference between two consecutive counting periods. For example, for a 30 mins step value and a 60 mins duration value, entry counts correspond to pictures taken every 30 mins, over a total duration of 60 mins.

Aircraft trajectories are defined as sequences of spatio-temporal points of type $(\text{long}_i, \text{lat}_i, \text{alt}_i, t_i)$, denoting the longitude, latitude and altitude of the aircraft, respectively, at specific time instances t_i. Casting them into a congestion resolution setting, trajectories may be represented as time series of events specifying resources used, i.e. sectors, 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 sector.

We consider the DCB problem to be an instantiation of the generic resources' congestion problem consisting of a finite set of discrete resources, such as segments of roads or railways, sea areas that vessels' trajectories cross, sectors in the airspace, communication network segments, logistics facilities, buffers or tools in a production facility. Each of these resources is related to a set of operational constraints, whose satisfaction is deemed necessary for agents to perform their tasks jointly. In this study we consider congestion problems related to the capacity of sectors (resources) and to demand–capacity imbalances that appear due to the demand of using the shared resources.

Specifically, in the DCB cases, each resource R_i is a sector with a specific capacity C_{R_i} and the goal is to resolve imbalances of demand and capacity of sectors. Cases where the demand is greater than maximum allowed capacity C_{R_i} are *capacity violated* or *demand–capacity imbalanced* cases. These are named as *hotspots* and result into *congestion problems*.

Agents are entities who demand the use of resources to perform their tasks. In the DCB case, an agent A_i is the aircraft performing a specific trajectory T_i, in a specific date and time. The problem is about agents to perform their trajectories jointly, using the required airspace in an efficient way w.r.t. resources' operational constraints, resolving any demand–capacity imbalances.

To resolve DCB problems, agents can modify their schedule of using resources by imposing a "*delay*" to the execution of their trajectories, i.e. agents may shift the *whole schedule* for using the required resources by a specific amount of time. Thus, agents have to learn joint delays to be imposed to their trajectories w.r.t. the operational constraints concerning the capacity of resources demanded.

We aim to resolve all hotspots (i.e. provide effective solutions of the DCB problem), minimising the total delay imposed to flights (i.e. the sum of delays to all flights), as well as the average delay (i.e. the ratio of total delay to the number of flights) w.r.t. the number of delayed flights.

Imposing different delays to trajectories may propagate the creation of congestion problems to different time periods and sectors. Agents that contribute to hotspots can be considered as "peers", since their decision affects the others and they have to jointly decide on their task delays. This implies that agents form "neighbourhoods" of interacting peers, allowing the exploitation of problem's spatial and temporal sparsity. The sets of interacting trajectories (i.e. the trajectories in a neighbourhood) may change due to the propagation and resolution of congestion problems. Thus, there is a necessity to dynamically update neighbourhoods of agents executing interacting trajectories, as agents decide on different delays.

We can use a graphical representation for modelling a society of agents $S = (\mathcal{T}, \mathcal{A}, \mathcal{E})$, where every vertex corresponds to a single agent A_i in $\mathcal{A}$ and any edge (A_i, A_j) in $\mathcal{E}$ connects agents with interacting tasks in $\mathcal{T}$. The procedure is dynamic since, edges' automatic update and new edges may be added in cases new interacting pairs of tasks (and thus agents) appear. We denote as $N(A_i)$ the *neighbourhood* of agent A_i, i.e. the set of agents (including itself) connected to agent $A_i \in \mathcal{A}$ which are its peers.

We use a range of MaxDelay_i time units (usually in minutes) as the available options for any agent A_i to delay its flight, i.e. $D_i = \{0, \ldots, \text{MaxDelay}_i\}$. These delays may also be assigned preferences that should be varied from 0 to MaxDelay_i. Note that the maximum preferred delay and the rate of decreasing preferences may differ among flights. Here we consider that all agents share the same *MaxDelay* and have no special preferences on delays, other than decreasing their own delay.

The DCB problem specification emphasises on the following problem aspects:

- Agents need to coordinate their strategies (i.e. chosen delays) to execute their tasks jointly with others, w.r.t. their preferences and operational constraints, so as to resolve all DCB problems occurring;
- Agents need to explore and discover how different combinations of available options affect the joint performance of their tasks w.r.t. the operational constraints, given that they do not know the interacting tasks that emerge due their joint decisions, and of course they do not know the new DCB problems;
- Agents' preferences on the options available may vary depending on the task performed, and are kept private;
- There are multiple and interdependent congestion problems that occur at the same time, in unpredictable ways for the agents who have to resolve them jointly;
- The setting is highly dynamic given that, to a great extent than state-of-the-art efforts, we consider interdependent congestion problems that change while agents choose their delay strategies.

4 Hierarchical Multiagent reinforcement learning for the DCB process

We formulate the DCB process of air traffic management as a multiagent reinforcement learning framework where the flight-agents operate in the same environment and share common resources. In this study we focus on hierarchical multiagent reinforcement learning schemes that construct multiple views of policies at multiple levels of abstraction. According to the problem specification we consider a hierarchical *Markov decision process* (MDP) that comprises the following constituents:

- The *society of agents* $S = (\mathcal{T}, \mathcal{A}, \mathcal{E})$, as specified above.
- A set of *abstraction levels* $L \in \{1, \ldots, h\}$.
- A *ground/abstract local state* per agent A_i at time t, comprising state variables that correspond to (a) the delay imposed to the trajectory T_i at time t executed by A_i denoted by $delay_i^{L,t}$. This ranges to the sets of options assumed by A_i in D_i, w.r.t the abstraction step at level L (defined below); (b) the number of hotspots in which A_i is involved in (for any of the sectors) at time step t denoted by $hotspots_i^t$. Such a local state is denoted by $s_i^{L,t}$. The *ground/abstract joint state* $\mathbf{s}_{ij}^{L,t}$ of agents A_i and A_j is the tuple of the ground/abstract state variables for both agents. The set of all ground/abstract joint states for any subset $N(A_i)$ of $\mathcal{A}$ (i.e. neighbourhood of agent A_i) is denoted $\mathbf{State}_{N(A_i)}^{L,t}$, and the set of ground / abstract global states is denoted by $\mathbf{State}^{L,t}$.
- A *set of actions* that is level and agent independent, and denoted by $\mathbf{Ac} = \{0, 1\}$. Such an *action* executed by agent A_i at time step t is denoted by a_i^t. In case the agent at a time point t is still on ground, may either increase its total delay, or not. The number of time instants (minutes) to be added, may vary, also depending on the abstraction step at any level. Thus, at each time point the agent has to take a binary decision, and maybe, a decision on the time instants to be added to its total delay. When the agent has taken off, then its strategy is considered fixed and it follows the intended/predicted trajectory. The *joint action* of a neighbourhood of agents $N(A_i)$ executing their trajectories at time t, is a tuple of local actions, denoted by $\mathbf{a}_{N(A_i)}^t$. The global action of all agents at any time instance is denoted as $\mathbf{a}^t$.
- An *(state) abstraction function* at every level L:

  $$\phi_L : s_i \rightarrow s_i^L$$

  that maps every ground local state of agent A_i (or ground joint state of agents A_i and A_j) to an abstract local state at level L, $s_i^{L,t}$ (respectively, $s_{ij}^{L,t}$)[2]. Specifically, the abstraction function ϕ_L maps ground states with respect to the abstraction step (specified below) applied on delays, to the corresponding abstract states, given that ground and abstract states have equal number of hotspots.
- The *abstraction step* at level $L \in \{1, \ldots h\}$, denoted as M_L, defining the amount of time instants that correspond to the same abstract time point, used in the state abstraction function. For example, when $M_L = 10$, then time instants 1-10 belong to the first abstract time point at level L, 11-20 to the second, etc. This is further discussed in Sect. 4.1. We consider that M_L decreases as we proceed from level h and moving towards level 1 and also that $M_1 = M_{ground} = 1$.
- The *state transition function Tr* at every level L, gives the transition to the global state $\mathbf{State}^{L,t+1}$ based on the global action $\mathbf{a}^t$, in global state $\mathbf{State}^{L,t}$:

[2] The mapping of joint states to abstract joint states is straightforward using ϕ_L, given that each joint state is a concatenation of local state parameters.

$(Tr)_L : \mathbf{State}^{L,t} \times \mathbf{a}^t \rightarrow \mathbf{State}^{L,t+1}$.

- The *local reward* of an agent A_i, denoted Rwd_i, is the reward that the agent gets by executing its local action in a local state at the ground level. Thus, the reward function is independent of the hierarchy level L, as it will be specified.
- A *(local) policy* of an agent A_i at level L, or at the ground level, is a function

  $\pi_i^L : s_i^{L,t} \rightarrow a_i^t$, or $\pi_i : s_i^t \rightarrow a_i^t$,

  respectively, that returns local actions for any given local state. The optimum policy should maximise the expected sum of future discounted rewards (called return), which is expressed by the state–action Q-value function, $Q_i^L(\phi_L(s_i^t), a_i^t)$. This describes the expected discounted reward received by starting from state $s_i^{L,t}$ at time instant t, executing the action a_i^t at t and following the policy π_i^L:

$$Q_i^{\pi_i^L}(\phi_L(s_i^t), a_i^t) = E_{\pi_i^L}\left[\sum_{k=0}^{\infty} \gamma^k \text{Rwd}_i^{t+k+1} | \pi_i^L\right] \quad (1)$$

  where γ is the discount factor range in [0, 1]. Estimating the optimal policy $(\pi_i^L)^*$ for agents is equivalent on choosing strategies that yield the best state–action value function, i.e.

$$(\pi_i^L)^* = \arg\max_{a_i} Q_i^{\pi_i^L}(\phi_L(s_i), a_i) \quad (2)$$

Within the reinforcement learning framework there are two possible forms of abstraction:

- *State–action abstraction*, which groups together states with similar environmental configurations and associated behaviour [4, 13, 23], and
- *Temporal abstraction*, which adds to the original action space abstraction layers of temporally extended action [22, 36, 42].

To solve these MDPs at different abstraction levels, we have devised the generic hierarchical multiagent reinforcement learning framework, which is presented in the following section.

4.1 Hierarchical multiagent reinforcement learning approaches

At first we present in detail the generic hierarchical Q-learning framework at multiple abstraction levels. As far as state abstraction is concerned, the presentation focuses only on delays of states which is the only state variable that is the subject of abstraction.

The proposed hierarchical Q-learning framework comprises the following phases:

- *Set abstract state at level L*, $\mathbf{State}^L$. The ground state space is partitioned into a number of K_L equidistant intervals of length equal to the abstraction step at level L, M_L. Thus, all ground states with delays between $d \times M_L$ and $(d+1) \times M_L$, $d = 0, 1, 2, \ldots$, assuming equal number of hotspots, are mapped to the same abstract state. I.e. $\phi_L(delay, hotspots) = (\lfloor delay/\ M_L \rfloor + 1, hotspots)$. An example of an abstract state construction procedure is shown in Fig. 1 that partitions the ground level ($L = 1$) of 40 min into $K = 8$ equidistant intervals ($M_2 = 5$) in the abstract level ($L = 2$).
- *Solve MDP at level L.* The agent A_i computes a policy π_i^L considering that state transitions happen only at the ground space, as specified in the MDP, i.e. at any time instant t the agent observes a ground local state s_i^t, maps it to the corresponding abstract local state $\phi_L(s_i^t) = s_i^{L,t}$ at level L and decides whether and how it will increase its delay.
- *Map solution from* $\mathbf{State}^L$ *to* $\mathbf{State}^{L-1}$ *(transfer learning).* After learning an abstract policy at level L the goal is to further refine it to a policy regarding $\mathbf{State}^{L-1}$. In this phase an initialisation of Q-values at level L-1 occurs in terms of the values learned at other levels, also w.r.t the actions per state. This is achieved according to the mapping between states in two successive levels, as indicated by the abstraction function.
- *Solve the MDP at the level* $L-1$. Given the Q^{L-1} values, a refinement phase in the abstract space $\mathbf{State}^{L-1}$ follows, towards computing a refined policy per agent at that level.

During our experiments we have considered $h = 2$ levels (one abstraction and the ground level). The advantages of the proposed state–action abstraction hierarchical scheme are as follows: (a) it provides flexibility to abstracting states and actions, (b) it allows tuning the initialisation strategy of the Q-learning at any level, given the results in the previous abstraction level, and c) it improves the capability to explore deeper the state–action space, supporting the discovery of better solutions in the original state–action space. Therefore, the hierarchical scheme at multiple levels allows the agents to have multiple policies: when $h = 2$, an abstract and a refined, possibly better than the abstract, policy. In our study we have used an abstraction step of size $M = 10$ (min).

Based on this generic framework, we have constructed four hierarchical learning reinforcement learning schemes that implement possible forms of abstraction:

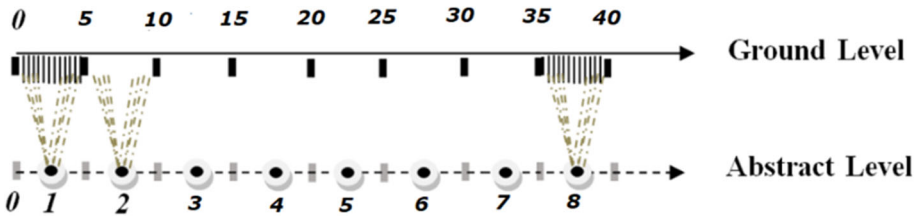


Fig. 1 An example of the construction of the abstract level for the state abstraction case: delay (max delay is 40 min) is partitioned into a number of $K = 8$ equidistant intervals of 5 min and consecutive ground level states are mapped to abstract level states

- State–action abstraction multiagent framework with independent reinforcement learners (*HMIRL*),
- State only abstraction multiagent framework with independent reinforcement learners (*sHMIRL*),
- Temporal (action) abstraction multiagent framework with independent reinforcement learners (*tHMIRL*),
- State–action abstraction collaborative environment of multiagent reinforcement learning (*HCMARL*).

4.1.1 Hierarchical multiagent reinforcement learning with independent learners (HMIRL)

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 [15] assumes that the global Q-function is a linear combination of local agent-dependent Q-functions.

Although the original proposal considers a single level of abstraction, in our case, each local Q-function Q_i^L w.r.t. abstraction level L for agent A_i, is calculated according to the local state at abstraction level L, as determined by the mapping function ϕ_L and the local action a_i. At any time point, the agent may increase its delay by adding M_L time instants, thus abstracting actions, in conjunction to states.

Based on the Q-learning update rule [23], when an agent observes a transition $(s_i^t, a_i^t, s_i^{t+1})$ at the ground level, it maps s_i^t to the corresponding abstract state and updates the Q-value $Q_i^L(\phi_L(s_i^t), a_i^t)$, w.r.t. abstraction level L as follows:

$$\begin{aligned} Q_i^L(\phi_L(s_i^t), a_i^t) = Q_i^L(\phi_L(s_i^t), a_i^t) \\ + \eta[\mathrm{Rwd}_i(s_i^t, a_i^t) + \gamma * max_{a_i'} Q_i^L \\ (\phi_L(s_i^{t+1}), a_i') - Q_i^L(\phi_L(s_i^t), a_i^t)] \end{aligned} \quad (3)$$

where η denotes the learning rate. It must be noted that (a) even though we update Q-values corresponding to abstract states, we compute the reward based on the ground state, and (b) instead of the global reward $\mathrm{Rwd}(\mathbf{s}, \mathbf{a})$ used in [15], we use the local reward Rwd_i received by the agent A_i.

4.1.2 State only abstraction multiagent framework with independent reinforcement learners (sHMIRL)

This method is a variation of the HMIRL method described above, doing state abstraction. It differs in three main aspects:

(a) It does not abstract actions: at any time point the agent may add one time instant (i.e. 1 min) to its total delay.

(b) When mapping solutions from $\mathbf{State}^L$ to $\mathbf{State}^{L-1}$: given a ground state s_i, the Q^L values for states $\phi(s_i) = s_i^L$ at any level of abstraction L and for any action are initialised to 0. During learning, these are updated independently, based on the HMIRL method specified above.

(c) When solving the MDP at the level $L-1$: given the Q^{L-1} values, this method exploits the estimation on the ground delay $delay\_estimation_i^L$ provided by solving the MDP at the level L to effectively limit the state space to be explored at level $L-1$. Specifically, the set of options assumed by A_i in $\mathbf{State}^{L-1}$ space is in $\{max\{0, delay\_estimation_i^L - d^{L-1}\}, ..., delay\_estimation_i^L\} \subseteq D_i$, where d^{L-1} is an amount of time instants we subtract from that delay decided at level L to provide some flexibility to the agent to refine its decision at level $L-1$.

4.2 Temporal abstraction multiagent reinforcement learning—tHMIRL

This is a temporal abstraction framework that forms a two-layer hierarchy of policies: a *meta-policy* and a *primitive policy*. In particular, as already defined in the generic framework, we decompose the policy of every agent into lower-level primitives or options, and a higher level meta-policy that triggers the appropriate behaviours: at first the agent decides its action, i.e. whether or not it requires additional time units of delay. If yes, differently from the other methods, it decides on the amount of time units to be added to its existing delay, by choosing from a set of alternatives. In our approach we have chosen three such

possible values $\{1, 3, 5\}$ (min). Therefore, the decided action a_i of every agent A_i comprises two aspects:

- $a_i^{(1)} = \{\text{True}, \text{False}\}$: This defines whether or not a delay will be added to the total delay of A_i flight (binary decision).
- $a_i^{(2)} = \{1, 3, 5\}$: This defines the number of minutes to be added to A_i flight delay, in case of $a_i^{(1)} = \text{True}$.

In this cases, we may consider that any agent learns two policies, jointly. Therefore, we use distinct Q-value functions as follows:

- $Q_i^1(\phi(s_i^t), a_i^{(1)})$ for the meta-policy, and
- $Q_i^2(\phi(s_i^t), a_i^{(2)})$ for the policy on the delay.

It must be noted that, during the learning process, both Q-values are updated based on Eq. 3, using the same reward function. As a result, it is possible to simultaneously have the Q-value updates at both levels for each decision step, when the meta-policy chooses "True". In the case $a_i^{(1)} = \text{False}$, only updates for $Q_i^1(\phi(s_i^t), a_i^{(1)})$ value function are provided.

4.2.1 Hierarchical collaborative multiagent reinforcement learning—HCMARL

The proposed multiagent reinforcement learning approach takes advantage of the problem structure (i.e. interactions among flights and problem factorisation), considering that agents do not know the transition model and interact concurrently with all of their peers.

It is a variant of a sparse cooperative Q-learning method proposed in [18]. Again, the original proposal considers a single level of abstraction, so in our case, each local Q-function Q_{ij}^L w.r.t. abstraction level L for the agents A_i and A_j, is calculated according to the joint state at abstraction level L, as determined by the mapping function ϕ_L and the joint action a_{ij}. In particular, assuming that two peer agents A_i and A_j are connected by an edge in the coordination graph, the joint-state Q-function for these agents is denoted as Q_{ij}^L.

According to the Q-learning algorithm the following update rule is obtained:

$$\begin{aligned} Q_{ij}^L(\phi_L(s_{ij}^t), a_{ij}^t) = (1-\eta) Q_{ij}^L(\phi_L(s_{ij}^t), a_{ij}^t) \\ + \eta \left[\frac{\text{Rwd}_i}{|N(A_i)|} + \frac{\text{Rwd}_j}{|N(A_j)|} \right. \\ \left. + \gamma * \max_{a'_{ij}} Q_{ij}^L(\phi_L(s_{ij}^{t+1}), a'_{ij}) \right] \end{aligned} \tag{4}$$

where η denotes the learning rate.

It must be noted that in the above rule the a'_{ij} is the best joint action of both agents A_i and A_j for the joint state s'_{ij}. In the literature this is estimated by the *max-plus* algorithm [18]. However, in our case and in order to reduce significantly the computational complexity of the update function, we have followed a simplified scheme, where we obtain the best actions directly from each peer agent's utility function:

$$a'_i = \arg\max_{a_i} Q_i^L(\phi_L(s_i), a_i) \tag{5}$$

$$a'_j = \arg\max_{a_j} Q_j^L(\phi_L(s_j), a_j) \tag{6}$$

4.3 Reward function

In many multiagent reinforcement learning problems, the task of determining the reward function in order to produce good performance is quite demanding. For the DCB problem we have formulated an individual delay reward Rwd_i for each agent A_i that depends on the participation (contribution) of that agent in hotspots occurring while executing its trajectory according to its decided delay. This is formulated with the following equation [41]:

$$\text{Rwd}_i(s_i^t, a_i^t) = C(s_i^t, a_i^t) - \lambda \times \text{DC}(a_i^t) \tag{7}$$

where

- $C(s_i^t, a_i^t)$ is a function that depends on the participation of agent A_i in hotspots while executing its trajectory according to its action a_i^t, and
- $\text{DC}(a_i^t)$ is a function related to delay cost of agent A_i.

The role of parameter λ is for balancing between the cost of participating in hotspots and the cost of the ground delay in an attempt to achieve the goal of zero hotspots and minimum delay.

Both functions represent delay costs at the strategic phase of operations. In particular, we have chosen the function $C(s_i^t, a_i^t)$ to depend on the total duration of the period in which agents fly over congested sectors, given by:

$$C(s_i^t, a_i^t) = \begin{cases} 81 \times \text{TDC}, & \text{if TDC} > 0 \\ C_+, & \text{if TDC} = 0 \end{cases} \tag{8}$$

where TDC is the total duration in congestions (i.e. hotspots in our problem case) for agent A_i. The condition of $\text{TDC} = 0$ holds when agents do not participate in hotspots and there is no congestion. In this case a large positive constant C_+ is received as reward. Finally, the coefficient 81 in the above rule is the average strategic delay cost per minute (in Euros) in Europe when 92% of the flights do not have delays [10].

The other function $\text{DC}(a_i^t)$ corresponds to the strategic delay cost when flights delay at gate. As suggested in [10],

this depends solely on the minutes of delay and the aircraft type. In our study we have used the next formulation:

$$\mathrm{DC}(a_i^t) = \mathrm{StrategicDelayCost}(a_i^t, \mathrm{Aircraft}(A_i)) \quad (9)$$

where *StrategicDelayCost*(.) is a function that returns the strategic delay cost given the features of a specific aircraft, $\mathrm{Aircraft}(A_i)$. Notice however that in the general case the $\mathrm{DC}(a_i^t)$ could be enriched with additional airline strategic policies and considerations regarding flight delays.

5 Simulated results

In order to measure the efficiency of the proposed approaches, we have created several evaluation cases of varying difficulty. Although the difficulty of DCB problems cannot be determined in a rigorous way, we followed an empirical study by inspecting problem parameters, taking also into account delays imposed by the Network Management (NM).

In particular, every evaluation case used in our study corresponds to a specific day above Spain during the year of 2016. The criteria that are considered for determining the level of their difficulty are:

- the number of flights involved,
- the average number of interacting flights per flight (i.e. 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 the NM,
- the average delay per flight for all flights according to NM, and
- the number of hotspots in relation to the number of flights participating in these hotspots.

Table 1 describes the evaluation cases consisting of the following quantities:

- *Number of flights*: The number of flights for that particular day above Spain;
- *Average traffic density*: The number of interacting flights (traffic) experienced by each of the agents (flights) in average;
- *Max delay*: The maximum delay imposed to any flight according to the NM;
- *Average delay*: The average delay per flight reported by the NM ignoring all delays with less than 4 min, according to experts advice;
- *Number of flights with delay*: The number of flights with delays due to imbalances, as reported by NM;
- *Max number of hotspots (number of flights)*: The initial number of hotspots together with the number of flights that participate to those hotspots (each flight may participate in multiple hotspots);

It must be noticed that while the NM specifies the delay to be imposed to each flight towards resolving demand–capacity imbalances, this is not a DCB problem solution: hotspots do occur even if NM delays are imposed to flights at the pre-tactical stage. This shows the tolerance of the system, as well its reliance to resolving imbalances in the tactical phase of operations, as opposed to the pre-tactical phase, according to our aim. Having said that, it is important to point out that delays imposed by the NM cannot be compared in a direct way to solutions provided by the proposed methods, given that, as already said, low predictability at the pre-tactical phase today, prohibits the NM to provide effective DCB problem solutions, and leaves decisions to be taken at the tactical phase. However, comparison shows the potential of reinforcement learning methods to solve problems effectively.

To construct an evaluation case initially we have to collect all planned flight trajectories (Flight Plans) as provided by the Spanish Operational Data. Based on the domain experts, we follow the Flight Plans specified just before take-off: This makes solutions provided by hierarchical MARL methods comparable to the delays imposed by the NM. All flights participating to the evaluation case are distinguished between commercial and non-commercial. It must be noted that delays cannot be imposed to non-commercial flights (e.g. military). The model of each

Table 1 Description of evaluation cases

Scenario	Evaluation cases			NM reported results		
	# Flights	Avg traffic density	Max delay	Avg delay	Regulated flights	# Hotspots (# flights)
Jul2	5572	6.39	80	1.663	498	29 (778)
Jul12	5408	5.84	95	0.95	254	28 (820)
Aug4	5544	6.41	66	0.383	146	33 (853)
Aug13	6000	10.89	147	1.152	415	53 (1460)
Sep3	5788	5.24	61	0.732	280	26 (783)

aircraft that executes a trajectory is stored for calculating the strategic delay costs.

Given any delay imposed to a trajectory, sectors crossed may vary due to the changing sector configurations. This may result into a number of alternative representations of a single trajectory; one for any possible delay, in the worst case. The planned trajectories (Flight Plans), in conjunction to the list of all the necessary sectors with their capacities, comprise an evaluation case. In addition, each evaluation case contains the following parameters:

- The number of flights (i.e. participating agents);
- The duration of the counting period for computing demand evolution (set to 60);
- The counting step for computing demand evolution (set to 30);
- The maximum possible delay (derived from the corresponding maximum delay imposed by the NM, as indicated in Table 1);
- The total duration H where in our study is 24 h;
- The learning rate η (was set to 0.01);
- The discount factor γ (was set to 0.99);
- The reward parameter λ (was experimentally set to 20).

At any level of any hierarchical multiagent reinforcement learning method we have considered a number of 15000 episodes following an ϵ-greedy exploration–exploitation strategy. In particular, initially we set the probability $\epsilon = 0.9$ and every 120 rounds we diminish it by the value of 0.01. To enhance the performance of the proposed methodology, we automatically set the flights which do not participate in any hotspot (i.e. agents with no neighbours) delay equal to zero (0) as a deterministic decision rule. However, it must be noted that any of these flights may participate in hotspots in the future, due to their joint strategies with the other agents during the multiagent reinforcement learning process.

Tables 2 and 3 present detailed comparative results of the four hierarchical MARL approaches, where we consider several statistical measurements that have been calculated after executing 10 independent experiments in any case. In particular we provide the mean value, the standard deviation (std), the median and the interquartile range (IRQ), for the average delay per flight and the number of regulated flights (i.e. flights with delay), respectively. The best mean value in each table and case is indicated in bold, while the second best is underlined. In addition, Fig. 2 illustrates the same results using box plots representations in the five evaluation cases. The first diagram (Fig. 2a) shows the average delay per flight, while the second one (Fig. 2b) presents the number of regulated flights. Notice that, following the standard practice in the domain, we do not consider delays less than 4 mins and the corresponding flights. In all cases the red coloured box plot corresponds to the HMIRL method, the green one to the sHMIRL method, the black one to the tHMIRL method, and the blue coloured box plot to the HCMARL method.

Table 2 Statistical measurements of the average delay per flight results, as calculated by 10 independent experiments

Scenario	Method	Mean	Std	Median	IRQ
Jul2	HMIRL	1.742	0.046	1.735	0.061
	sHMIRL	**1.358**	0.054	1.350	0.005
	tHMIRL	1.637	0.060	1.650	0.068
	HCMARL	<u>1.590</u>	0.049	1.590	0.080
Ju112	HMIRL	0.410	0.028	0.408	0.033
	sHMIRL	<u>0.187</u>	0.015	0.185	0.028
	tHMIRL	0.205	0.016	0.205	0.020
	HCMARL	**0.103**	0.009	0.100	0.010
Aug4	HMIRL	1.129	0.056	1.130	0.055
	sHMIRL	**0.731**	0.028	0.730	0.035
	tHMIRL	0.846	0.042	0.855	0.065
	HCMARL	<u>0.783</u>	0.056	0.780	0.090
Aug13	HMIRL	1.168	0.041	1.159	0.053
	sHMIRL	<u>0.996</u>	0.044	0.990	0.075
	tHMIRL	**0.975**	0.042	0.975	0.048
	HCMARL	1.115	0.053	1.110	0.035
Sep3	HMIRL	0.855	0.064	0.867	0.104
	sHMIRL	**0.578**	0.038	0.570	0.038
	tHMIRL	0.896	0.045	0.895	0.028
	HCMARL	<u>0.790</u>	0.041	0.780	0.065

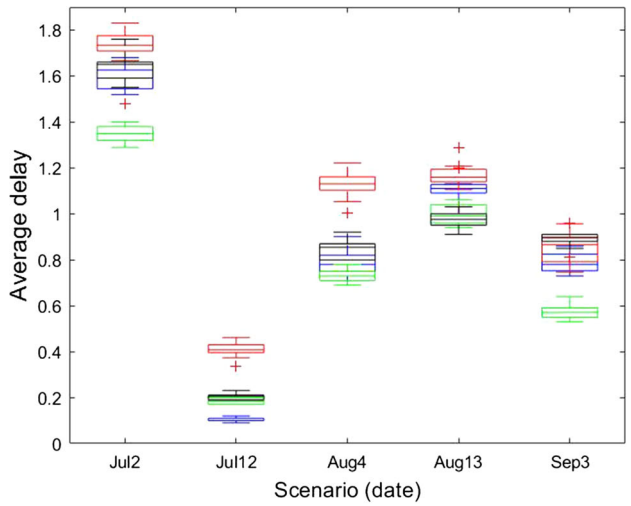
The best mean value is indicated in bold, and the second best is underlined

The sHMIRL and HCMARL seems to be the dominant methods as they prevail in terms of average delay in almost all of the cases. Only in the case of *Aug*13 the tHMIRL presents the best average delay per flight, with sHMIRL presenting the second best result, but HCMARL presents the lowest number of regulated flights. Comparing our methods with the results provided by the NM (see Table 1), we observe that these methods consistently provide better performance, except in the case of *Aug*4 where all methods are unable to outperform the NM. However, as already pointed out, delays imposed by the NM do not resolve all the imbalances, as the proposed methods do. As an example, the NM delays resolve only 2 hotspot occurrences out of 33 in *Aug*4 scenario (see Table 1). The reported results of our methods in conjunction with the delays imposed by the NM, show the effectiveness of the proposed hierarchical learning methods and their ability to provide qualitative solutions to real-world complex problems.

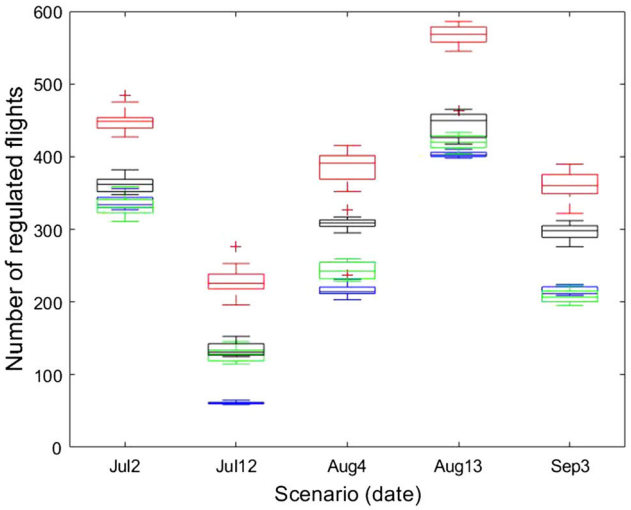
Table 3 Statistical measurements of the regulated flights results as calculated by 10 independent experiments

Scenario	Method	Mean	Std	Median	IRQ
Jul2	HMIRL	448.70	13.75	448.50	14.25
	sHMIRL	**331.90**	14.74	329.50	16.25
	tHMIRL	361.20	10.40	362.00	15.00
	HCMARL	337.00	8.56	334.00	12.50
Ju112	HMIRL	228.65	17.66	225.50	19.75
	sHMIRL	127.90	9.45	128.50	13.25
	tHMIRL	135.00	9.37	131.00	13.75
	HCMARL	**61.20**	1.61	61.00	2.00
Aug4	HMIRL	385.75	20.19	391.00	30.75
	sHMIRL	243.00	12.11	242.00	21.75
	tHMIRL	309.00	9.07	309.00	8.00
	HCMARL	**216.40**	8.30	214.00	8.50
Aug13	HMIRL	566.50	12.97	568.00	19.50
	sHMIRL	421.70	17.34	419.00	14.75
	tHMIRL	443.30	18.87	449.50	31.50
	HCMARL	**402.81**	3.71	402.00	5.00
Sep3	HMIRL	359.45	17.31	360.00	25.25
	sHMIRL	**207.90**	9.02	206.50	13.00
	tHMIRL	296.90	11.11	298.00	13.25
	HCMARL	216.20	5.10	215.00	9.00

The best mean value is indicated in bold, and the second best is underlined



(a) Average delay per flight, and per scenario



(b) Number of regulated flights per scenario

Fig. 2 Comparative results in terms of **a** the average delay per flight and **b** the number of regulated flights presented in box plots for the four comparative hierarchical methods per evaluation case: HMIRL (red), sHMIRL (green), tHMIRL (black) and HCMARL (blue)

The advantages of both hierarchical schemes, sHMIRL and HCMARL, are

- The capability to effectively explore the state–action space at different levels of abstraction. This is due to a more informative initialisation of the original (ground) space by the abstract layer, which allows to enhance the learning procedure and discover more optimal solutions.
- Provide with transfer learning mechanisms through layers exploiting and combining multiple policies regarding the DCB problem.

Furthermore, comparing these results to those reported by non-hierarchical methods for the same evaluation cases in [20], it is clear that the hierarchical framework improves their performance and increases their effectiveness.

We can also conclude from the constructed box plot diagrams the robustness of the sHMIRL and HCMARL methods. Figure 2 shows that in both measures, average delay and regulated flights, the sHMIRL and HCMARL methods present the best results among the hierarchical methods, with sHMIRL prevailing in most of the cases. The other methods either do not compute effective results, or present several outliers among the results. More specifically, HMIRL presents outliers for the average delay in all cases, and for the regulated flights the outliers are present in all cases except *Sep*3. tHMIRL on the other hand does not manage to report results that are better than those provided by sHMIRL or HCMARL. This is due to the complexity of the learning process (the agent has to decide on both, whether it will increase its delay and how much). On the other hand, this phenomenon with outliers does not appear in the case of the HCMARL, tHMIRL and sHMIRL methods, where the standard deviation and the interquartile range are lower in almost every evaluation case, as shown in Tables 2, 3.

6 Conclusions

In this study we introduce a hierarchical MDP and a hierarchical multiagent reinforcement learning framework that can take advantage of state or action abstractions, at any level of abstraction. This framework leaves open several options, and these corresponding (a) to the way state–action values are initialised at any level of abstraction, given the values of the previous levels, (b) whether agents take advantage of a coordination graph, and thus taking into account interdependencies among their actions, or whether they learn and act totally independently, (c) the levels of abstractions considered, and (d) whether each level takes into account the solution provided by the previous level to restrict the exploration space towards providing a refined and better solution.

Exploring these alternatives, we provided a set of methods working at two levels of abstraction (an abstract and a ground one). These methods have been applied for solving the multiagent MDP problem for resolving demand–capacity imbalances during the pre-tactical phase in the air traffic management domain. The effectiveness of the proposed methods has been demonstrated on real-world cases that encompasses a large number of agents and complex congestion settings. Among the proposed methods, evaluation results showed that the hierarchical collaborative method as well as the hierarchical independent learners method, where state–action values computed at previous levels are not transferred to the other levels and where the search space at each level is restricted to a space close to the solution provided by the previous level, are the most effective methods showing consistency of behaviour among the cases.

Since the results we took were very promising, it is our intention to further pursue and develop the methods in two main directions: at first we can employ value function approximation reinforcement learning schemes assuming continuous-values state spaces and examine the possibility of extending to a deep RL framework. Second, we aim to study alternative reward function schemes—taking also into account state-of-the-art reward schemes used in multiagent congestion problems, encompassing more features of the DCB problem in ATM.

Acknowledgements This work has been partially supported by the National Matching Funds 2017-2018 of the Greek Government and more specifically by the General Secretariat for Research and Technology (GSRT), related to DART(www.dart-research.eu) project. The major part of this work has been completed during the DART project where authors participated as members of the University of Piraeus Research Center group. We would like also to acknowledge the contribution of CRIDA for providing the data during the DART project.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Abel D, Hershkowitz DE, Littman ML (2016) Near optimal behavior via approximate state abstraction. In: International conference on machine learning (ICML'16), vol 48, pp 2915–2923
2. Agogino AK, Tumer K (2012) A multiagent approach to managing air traffic flow. Auton Agents Multiagent Syst 24:1–25
3. Andreas J, Klein D, Levine S (2017) Modular multitask reinforcement learning with policy sketches. In: 34th international conference on machine learning (ICML), pp 166–1751
4. Andrienko G, Andrienko N, Bak P, Keim D, Wrobel S (2013) Visual analytics of movement. Springer, Berlin
5. Bacon P, Harb J, Precup D (2017) The option-critic architecture. In: Proceedings of the thirty-first AAAI conference on artificial intelligence, AAAI'17, pp 1726–1734. AAAI Press
6. Bai A, Srivastava S, Russell S (2016) Markovian state and action abstractions for MDPs via hierarchical MCTS. In: Proceedings of the twenty-fifth international joint conference on artificial intelligence, IJCAI'16, pp 3029–3037. AAAI Press
7. Bazzan ALC, Wahle J, Klügl F (1999) Agents in traffic modelling—from reactive to social behaviour. In: 23rd annual german conference on artificial intelligence, pp 303–306
8. Chen J, Wang Z, Tomizuka M (2018) Deep hierarchical reinforcement learning for autonomous driving with distinct behaviors. In: 2018 IEEE intelligent vehicles symposium (IV), pp 1239–1244
9. Colby M, Tumer K (2013) Multiagent reinforcement learnng in a distributed sensor network with indirect feedback. In: International conference on autonomous agents and multi-agent systems (AAMAS'13), pp 941–948
10. Cook AJ, Tanner G (2015) European airline delay cost reference values. http://www.eurocontrol.int/publications/european-airline-delaycost-reference-values
11. Dayan P, Hinton GE (1992) Feudal reinforcement learning. In: Advances in neural information processing systems, [NIPS Conference], vol 5, pp 271–278
12. Delvin S, Yliniemi L, Kudenko D, Tumer K (2014) Potential-based difference rewars for multiagent reinforcement learning. In: International conference on autonomous agents and multi-agent systems (AAMAS'14), pp 165–172
13. Dietterich T (2000) Hierarchical reinforcement learning with the maxq value function decomposition. J Artif Intell Res 13:227–303
14. Frans K, Ho J, Chen X, Abbeel P, Schulman J (2017) Meta learning shared hierarchies. Technical Report. arXiv preprint arXiv:1710.09767
15. Guestrin C, Lagoudakis M, Parr R (2002) Coordinated reinforcement learning. In: International conference on machine learning (ICML'02), pp 227–234
16. Jong N, Stone P (2005) State abstraction discovery from irrelevant state variables. In: International joint conference on artificial intelligence (IJCAI '05), pp 752–757
17. Karp R, Koutsoupias E, Papadimitriou C, Shenker S (2000) Optimization problems in congestion control. In: 16th Annual symposium on foundations of computer science, pp 66–74
18. Kok JR, Vlassis N (2006) Collaborative multiagent reinforcement learning by payoff propagation. J Mach Learn Res 7:1789–1828

19. Konidaris G, Barto A (2009) Efficient skill learning using abstraction selection. In: International joint conference on artificial intelligence (IJCAI '09), pp 1107–1112
20. Kravaris T, Spatharis C, Bastas A, Vouros GA, Blekas K, Andrienko G, Andrienko N, Garcia JM (2019) Resolving congestions in the air traffic management domain via multiagent reinforcement learning methods. Technical Report. arXiv preprint arXiv:1912.06860
21. Kravaris T, Vouros G, Spatharis C, Blekas K, Chalkiadakis G (2017) Learning policies for resolving demand–capacity imbalances during pre-tactical air traffic management. In: Multiagent system technologies—15th German conference (MATES'17), pp 238–255
22. Kulkarni T, Narasimhan K, Saeedi A, Tenenbaum J (2016) Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In: Advances in Neural Information Processing Systems (NIPS'16), pp 3675–3683
23. Li L, Walsh T, Littman M (2006) Towards a unified theory of state abstraction for MDPs. In: International symposium on artificial intelligence and mathematics (ISAIM'06)
24. Ma A, Ouimet M, Cortés J (2020) Hierarchical reinforcement learning via dynamic subspace search for multi-agent planning. Auton Robot 44:485–503. https://doi.org/10.1007/s10514-019-09871-2
25. Makar R, Mahadevan S, Ghavamzadeh M (2001) Hierarchical multi-agent reinforcement learning. In: Proceedings of the fifth international conference on autonomous agents, AGENTS'01, pp 246–253
26. Malialis K, Delvin S, Kudenko D (2016) Resource abstraction for reinforcement learning in multiagent congestion problems. In: International conference on autonomous agents and multi-agent systems (AAMAS'16), pp 503–511
27. Mannor S, Menanche I, Hoze A, Klein U (2004) Dynamic abstraction in reinforcement learning via clustering. In: International conference on machine learning (ICML'04). https://doi.org/10.1145/1015330.1015355
28. McGovern A, Barto A (2001) Automatic discovery of subgoals in reinforcement learning using diverse density. In: Proceedings of the eighteenth international conference on machine learning (ICML'01), pp 361–368
29. Meyers C (2006) Network flow problems and congestion games: complexity and approximation results. Ph.D. thesis, MIT
30. Milchtaich I (2004) Social optimality and cooperation in nonatomic congestion games. J Econ Theory 114(1):56–87
31. Nachum T, Gu SS, Lee H, Levine S (2018) Data-efficient hierarchical reinforcement learning. In: 32nd Conference on neural information processing systems (NeurIPS 2018), pp 3303–3313
32. Parr R, Russell S (1998) Reinforcement learning with hierarchies of machines. In: Proceedings of the 1997 conference on advances in neural information processing systems (NIPS'97), vol 10, pp 1043–1049
33. Peng XB, Berseth G, Yin K, van de Panne M (2017) Deeploco: dynamic locomotion skills using hierarchical deep reinforcement learning. ACM Trans Graph 36(4):1–13. https://doi.org/10.1145/3072959.3073602
34. Penn M, Polukarov M, Tennenholtz M (2011) Congestion games with failures. Discr Appl Math 159(15):1508–1525
35. Radulescu R, Vrancx P, Nowe A (2017) Analysing congestion problems in multi-agent reinforcement learning. In: Proceedings of the 16th conference on autonomous agents and multiagent systems (AAMAS'17), pp 1705–1707
36. Rasmussen D, Voelker A, Eliasmith C (2017) A neural model of hierarchical reinforcement learning. PLoS One 12(7):e0180234. https://doi.org/10.1371/journal.pone.0180234
37. Riemer M, Liu M, Tesauro G (2018) Learning abstract options. In: Proceedings of the 32nd international conference on neural information processing systems, NIPS'18, pp 10445-10455. Curran Associates Inc., Red Hook
38. Rosenthal RW (1973) A class of games processing pure-strategy nash equilibria. Int J Game Theory 2:65–67
39. Spatharis C, Blekas K, Bastas A, Kravaris T, Vouros GA (2019) Collaborative multiagent reinforcement learning schemes for air traffic management. In: 10th international conference on information, intelligence, systems and applications (IISA), pp 1–8
40. Spatharis C, Kravaris T, Vouros GA, Blekas K, Chalkadiakis G, Garcia JMC, Fernández EC (2018) Multiagent reinforcement learning methods to resolve demand capacity balance problems. In: Hellenic A.I. conference (SETN 2018), pp 2:1–2:9
41. Spatharis C, Kravaris T, Vouros GA, Blekas K, Cordero JMG (2018) Multiagent reinforcement learning methods for resolving demand—capacity imbalances. In: Digital avionics systems conference (DASC'18)
42. Sutton R, Precup D, Singh S (1999) Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning. Artif Intell 112(1–2):181–211
43. Tessler C, Givony S, Zahavy T, Mankowitz DJ, Mannor S (2017) A deep hierarchical approach to lifelong learning in minecraft. In: Proceedings of the thirty-first AAAI conference on artificial intelligence (AAAI '17), pp 1553–1561
44. Tumer K, Welch Z, Agogino A (2008) Aligning social welfare and agent preferences to alleviate traffic congestion. In: Proceedings of the 7th international joint conference on autonomous agents and multiagent systems (AAMAS'08), vol 2, pp 655–662