DOI: 10.13009/EUCASS2023-713
Aerospace Europe Conference 2023 – 10TH EUCASS – 9TH CEAS

Single and Multi-Agent Reinforcement Learning Approach to Optimize Aircraft Ground Trajectories at Airports

Maxime Szymanski, Georges Ghazi†, and Ruxandra M. Botez†
École de technologie supérieure (ÉTS)
Laboratory of Applied Research in Active Controls, Avionics, and AeroServoElasticity (LARCASE)
Montreal, QC, Canada
georges.ghazi@etsmtl.ca · ruxandra.botez@etsmtl.ca
†Corresponding author

Abstract

This paper presents a methodology for routing aircraft at airports using a reinforcement learning approach. For this purpose, an aircraft was considered as an agent, while the airport in which the aircraft moves was considered as the environment. The airport was modeled as an undirected graph, composed of nodes (taxiway junctions) and segments (taxiways). The objective of the agent was to learn how to move from a departure node to a destination node by taking an optimal path. The optimal path within the scope of this study was assumed to be the shortest path, avoiding sharp turns (i.e., turns over 90°) while limiting the number of turns along the way. The agent was trained using the Proximal Policy Optimization (PPO) algorithm by considering a single-agent environment. The trained agent was tested for different airports and over 400 different scenarios were designed by randomly selecting a departure node and a destination node. The results showed that the agent was able to find the optimal path with an average success rate of 98%. Then, the agent was tested in a multi-agent environment by considering multi-aircraft moving in the same airport. Similar results were obtained. The results obtained in this study have shown that reinforcement learning methods can provide a very good solution to the problem of aircraft routing at airports.

1. Introduction

Airports are essential nodes in the aviation ecosystem, connecting countries and people worldwide. However, with air traffic expected to double to 8.2 billion by 2037 [1], this ecosystem may experience negative changes over the next decade. Indeed, most major airports are currently operating near their maximum capacity, and an increase in the number of aircraft will significantly amplify their demand for runway and terminal capacity, leading to congestion and delays [2]. This congestion will result in extended taxiing times, longer turnaround periods for aircraft, and reduced scheduling flexibility. Consequently, the overall efficiency of airport operations may be compromised, impeding the ability to accommodate additional flights, and limiting the potential for growth.

Airport congestion not only causes operational problems, but also adversely affects the environment. Indeed, increasing the number of aircraft will inevitably lead to excessive fuel consumption. This problem is even more critical at airports, since aircraft engines are designed to operate efficiently at cruising speeds and altitudes, whereas their performance is not optimal during low-speed ground operations. Taxiing at congested airports accounts for 6% of aircraft fuel consumption for short-haul flights. By burning fuel, aircraft engines release substantial quantities of pollutants, such as carbon dioxide (CO_2), nitrogen oxides (NO_x), carbon monoxide (CO), particulate matter (PM), and volatile organic compounds (VOCs), which contribute to the deterioration of local air quality [3] These emissions also lead to the formation of ground-level ozone, which has harmful effects on human health [4].

Recognizing this dual operational and environmental challenge, efforts are being made to optimize aircraft ground operations by exploring innovative strategies and technologies, such as advanced air traffic management systems and intelligent ground operations, with the aim of minimizing fuel consumption and associated emissions. Within this context, it becomes imperative to investigate new solutions to revolutionize traffic management and move towards a more flexible and sustainable air transport system.

DOI: 10.13009/EUCASS2023-713

In recent years, extensive research has been conducted to propose new efficient flight procedures (i.e., above 1500 ft). Continuous Descent Operations (CDOs) are one of the most notable examples of how fuel consumption and associated emissions can be reduced by improving flight procedures. Unlike the Standard Step-Down Approach (SDA), CDO enables aircraft to descend continuously with idle thrust setting while avoiding level-off segments [5]. In this way, CDO allows for a smoother and more efficient approach to landing, ultimately leading to a more sustainable and cost-effective operation for airlines [6]. Another promising approach to improve flight efficiency is based on flight trajectory optimization. By considering various factors, such as aircraft performance and atmospheric conditions, airlines can determine the most efficient trajectory for a given flight using different optimization techniques, which can result in reduced fuel consumption, which not only lowers operating costs but also reduces aircraft emissions. The most common way to improve flight efficiency is to optimize the speed [7, 8] and the altitude [9, 10] during the cruise phase, using highly accurate models [11–13]. However, with the intention of moving towards a more flexible airspace, recent research has seen a trend in the concept of free flight. The concept of free flight refers to a paradigm shift in air traffic management aimed at providing aircraft with greater flexibility and autonomy in their flight paths. By reducing or eliminating the traditional dependency on fixed airways and predefined routes, aircraft can navigate more freely within the airspace while maintaining their safety and efficiency. These concepts paved the way for the development of new flight procedures aimed at optimizing the vertical [14], the lateral [15, 16], or a combination of both [17–19] profiles, leading to a more flexible, dynamic and cost-effective air transport system.

The concept of free flight has attracted considerable attention in the air traffic management community where research primarily focused on airborne flight phases. However, extending this concept to airport environments offers the potential for addressing and improving the challenges associated with airport ground movement. Indeed, transposing free flight principles to ground operations enables airports to explore innovative approaches to enhance the efficiency, safety, and overall management of aircraft movement on taxiways and runways. This problem, known as Airport Ground Movement (AGM) optimization, involves moving an aircraft from a gate to a runway (or vice versa) within a given time window while minimizing fuel consumption and complying with safety constraints to avoid conflicts with other aircraft. From an optimization point of view, this problem can be seen as a combination of routing and scheduling problems, such as the gate/stand allocation problem [20, 21], and runway sequencing problems [22, 23].

For small- and medium-sized airports with low traffic density, routing aircraft between gates and runways can be addressed using typical shortest-path algorithms. Earlier studies in this field were conducted by Gotteland and Durand [24], who used the Branch & Bound algorithm to generate a list of predetermined shortest routes between different points for a given airport. The authors then used a genetic algorithm to assign, from the generated list, the most appropriate route and waypoints for each aircraft. Similarly, Brownlee *et al.* [25] used an A* algorithm to determine optimal ground paths that minimize taxi time. They combined their shortest taxi-time algorithm with a rolling window approach and a genetic algorithm to optimize aircraft allocation order. Dabachine *et al.* [26] applied and compared three routing algorithms, including Dijkstra, bidirectional Dijkstra, and A*. According to the results, the Dijkstra variants were able to find the optimal solution, but the cost in computation time remained relatively high. Conversely, the A* was able to find a solution faster, but its convergence did not guarantee that the optimal solution was always obtained. A hybrid A* algorithm was then proposed to take advantage of each algorithm to reduce computation time and optimize the search for the shortest path. The A* algorithm was also explored by Lesire [27] and Zhou and Jiang [28] for optimizing aircraft paths while minimizing conflict probability. In [29], Ravizza *et al.* proposed a new algorithm called the QPPTW (Quickest Path Problem with Time Window) to solve the multi-aircraft routing problem. The algorithm proposed in this study was designed based on the Dijkstra algorithm by incorporating a segment reservation principle imposed by previously routed aircraft. This approach enabled the routing of multiple aircraft while guaranteeing the absence of conflicts by ensuring that two aircraft do not occupy the same route segment.

One of the drawbacks of shortest-path algorithms, such as those applied in the previous studies, is that they cannot optimize several objectives simultaneously. In most cases, an optimization algorithm is required to find the optimal aircraft path, while other constraints or objectives must be incorporated by modifying the nature of the optimization problem. As a result, these methods are limited to optimizing one trajectory at a time and do not guarantee to find all optimal solutions for multiple aircraft routing problems. Another solution would be to run these algorithms several times for each aircraft, but this approach is time-consuming, and convergence is not necessarily guaranteed.

In recent years, Artificial Intelligence (AI) has revolutionized trajectory planning in robotics by enabling intelligent decision-making and adaptive behavior. Using AI algorithms, and more specifically Reinforcement Learning (RL) algorithms, robots can be trained to learn through trial and error how to dynamically select optimal trajectories based on a set of target goals in an environment of dynamic constraints. This iterative learning process allows robots to adapt

DOI: 10.13009/EUCASS2023-713

to changing environments, avoid obstacles, and navigate complex terrain efficiently. Within this context, it would be interesting to transpose these concepts to solve the airport ground movement problem. In this case, an aircraft can be considered as a robot, while the airport can be considered as the environment in which the aircraft must find the most efficient route from a departure node (i.e., gate) to an arrival node (i.e., runway entry point), while avoiding conflicts with other aircraft. The main objective of this paper is, therefore, to present a study conducted at the Laboratory of Applied Research in Active Controls, Avionics, and AeroServoElasticity (LARCASE) to investigate the possibility of using Reinforcement Learning for solving the routing aircraft problem. To this end, the airport was modeled as an undirected graph, where the edges represent taxiways, and the nodes represent intermediate junctions. Aircraft were trained to move from one node to another by selecting the node that leads to the optimal trajectory based on a probabilistic score. As a first step, the optimal trajectory in the context of this study was assumed to be the shortest path. However, this concept of aircraft routing based on a probabilistic score could be generalized to include other constraints such as conflict probability, fuel consumption, or emissions.

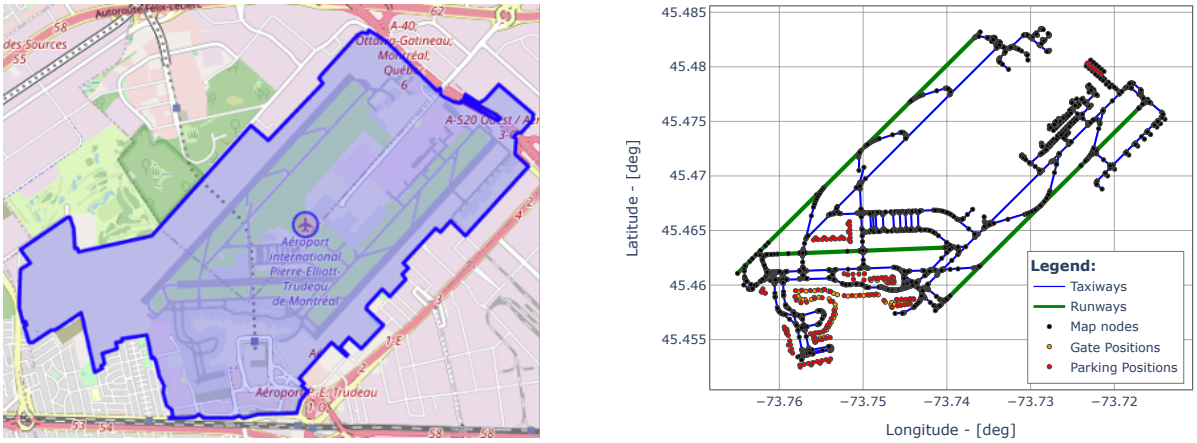
This paper is organized as follows: **Section 2** introduces general concepts such as airport representation and the concept of reinforcement learning. **Section 3** deals with the various steps of the proposed methodology. **Section 4** presents the results obtained and an example of an application with several aircraft at Montreal Airport. Finally, the paper ends with conclusions and remarks on future work.

2. Airport Layout and Background

Before presenting the methodology to manage aircraft ground trajectories between departure and arrival points, it is necessary to introduce several notations and fundamental concepts. For this purpose, this section first presents the approach considered for modeling an airport and its main components, and includes collecting cartographic data and generating a graph reflecting the airport topology. Subsequently, the section focuses on the concept of reinforcement learning.

2.1 Airport Layout Modelling

For modeling purposes, an airport is represented in this study by an undirected graph consisting of a set of nodes connected by edges. As most airports have relatively complex topologies, manually creating a graph to reproduce their structure can be challenging. Thus, to simplify this aspect, a tool was developed in a previous study at the LARCASE laboratory [30] to automatically generate an airport graph using mapping information from the OpenStreetMap (OSM) website[1]. The OSM is a collaborative project to create a free and editable cartographic database worldwide. This database was used to collect all the elements that constitute an airport, such as runways, taxiways, gates, and parking positions.



(a) Airport Representation in OSM

(b) Airport Graph Layout

Figure 1: Representation of Montreal Trudeau International Airport (Montreal, QC, Canada)

Figure 1a shows a typical example of a map representation of Montreal Airport (CYUL, Trudeau International Airport) obtained from OSM, where the airport area is highlighted in blue.

[1] https://www.openstreetmap.org/

DOI: 10.13009/EUCASS2023-713

All OSM data located in the blue area in Fig. 1a can be exported into an *.xml file. After downloading the *.xml file, a graph of the airport can be constructed by retrieving all nodes belonging to a runway or a taxiway. A connectivity (or adjacency) matrix is then defined. This connectivity matrix is the mathematical representation of the airport layout, and is composed of 0 and 1, where 1 indicates that two nodes are connected by a taxiway or runway (i.e., a segment). All other nodes corresponding to gates, parking, or holding positions are also saved, as they may be required for detailed analyses. Figure 1b shows the result of the graph layout generated for the Montreal Airport (CYUL).

Once the graph has been designed, attributes are associated with each segment that composes it. An attribute is a characteristic that qualifies a segment (i.e., a portion of a taxiway or runway). The attributes considered in this study for a segment are the way type (i.e., runway or taxiway), the reference name, the length, the bearing, and the speed limit. It should be noted that speed limits may change depending on airports, airlines, or aircraft manufacturer recommendations. For simplicity, it was assumed that an aircraft should not exceed 30 kts on straight segments while turning segments were restricted with a maximum turning speed of 10 kts. These values are user-defined parameters and can be modified to estimate better the taxi time between a departure node and an arrival node.

2.2 Reinforcement Learning

An optimization problem can be very difficult to solve, especially when there are no data sets to model the cost function to be optimized, or when the cost function is difficult to quantify mathematically. Reinforcement Learning (RL) is a branch of Artificial Intelligence (AI) that uses the art of "*learning by trying*" to solve complex optimization problems. This approach is mainly based on an iterative process in which an agent learns to interact with a dynamic environment in order to improve its actions through trial and error and using feedback from the environment.

Figure 2 schematically illustrates the action-feedback principle of a typical RL model [31].

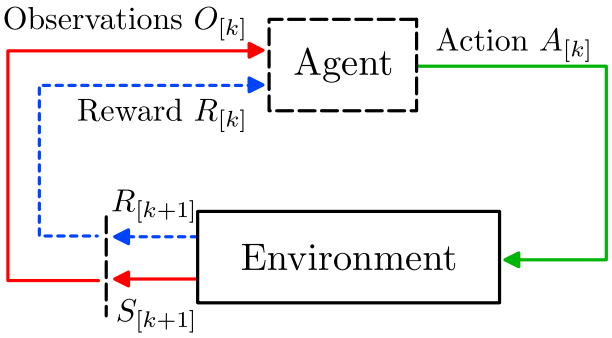


Figure 2: Generic Reinforcement Learning (RL) Model

The model shown in Fig. 2 can be described as a Markov decision process involving two main elements: the environment and the agent. The environment is a mathematical representation of the physical world in which the agent evolves, while the agent is a system that can act on the environment in order to modify its characteristics or state.

The interactions between the agent and the environment are done in discrete steps. At a given iteration (or step) k, the agent receives a set of observations $O_{[k]}$, reflecting certain characteristics of the environment, and a reward $R_{[k]}$. Based on these two elements, the agent performs an action $A_{[k]}$. The aim of this action is to change the environment state $S_{[k]}$ to a new state $S_{[k+1]}$. A new reward $R_{[k+1]}$ associated with the transition $S_{[k]} \rightarrow S_{[k+1]}$ is then determined using a "*reward function*". This reward represents the degree of effectiveness of the agent action. Basically, the more effective the action, the higher the reward. The process is then repeated with these new observations and the new reward over several iterations, which constitute an episode.

Depending on the complexity of the problem, the agent can be modeled by one or more neural networks. The neural network receives the observations $O_{[k]}$ and the reward $R_{[k]}$ as inputs, processes the data, and then generates an action $A_{[k]}$ as an output. This operation defines the "*agent policy*" and depends on the weights of the neural network, which are updated at each iteration based on the collected rewards. The agent policy performance is highly dependent on the values of the neural network weights. Thus, the objective of a reinforcement learning problem is to optimize the weights of the neural network to enable the agent to learn an optimal policy that maximizes the expected cumulative reward over the iterations.

DOI: 10.13009/EUCASS2023-713

Proximal Policy Optimization

The Proximal Policy Optimization (PPO) is a model-free policy gradient learning algorithm proposed by Schulman *et al.* [32] to train an agent. This algorithm was introduced by OpenAI to overcome the limitations observed in conventional policy gradient algorithms by providing a more stable and efficient approach during the training process. For this purpose, the algorithm alternates between sampling data through environmental interaction and optimizing a surrogate objective function using stochastic gradient descent. This aspect enhances training stability by constraining the size of the policy change at each step of the optimization of the weights of the agent neural network, thus ensuring a controlled and gradual evolution of the policy.

The PPO is a stochastic algorithm, wherein the decision-making process of the agent relies on a probability distribution rather than a deterministic value. This feature improves the exploration of the environment by enabling the selection of sub-optimal actions. In addition, the PPO algorithm uses an "*on-policy*" approach, meaning that it adjusts the agent policy based on previous policies, thereby effectively leveraging past actions to enhance its convergence properties.

When using the PPO algorithm, the agent must be characterized by two neural networks, comprising an actor-network and a critic-network. The actor-network, denoted as $\pi\left(A_{[k]}|O_{[k]};\theta\right)$, receives the input observations, and generates the probability of taking actions based on the weights θ. Conversely, the critic-network, denoted as $V\left(A_{[k]}|O_{[k]};\phi\right)$, receives the input observations, and predicts the anticipated long-term reward associated with the specific state under consideration based on the weights ϕ.

The use of actor-critic architecture has several advantages over simple neural network architecture. The first advantage is its stability. Indeed, the actor-critic architecture stabilizes the learning process by decoupling exploration policy (actor) from reward estimation (critic). This decoupling prevents policy changes from affecting reward estimates, resulting in a more stable process. Another advantage is that this dual architecture enables better convergence, as the critic provides the agent with an estimate of the quality of the action, thus helping it to take the right decision. In this case, the actor can use the estimate and the actual reward to modify and adapt better its policy.

Multi-Agent Reinforcement Learning

A Multi-Agent Reinforcement Learning (MARL) model is an extension of a Single-Agent Reinforcement Learning (SARL) model, in which many agents act in a shared environment and have partial or complete knowledge of the state or actions of other agents. In addition, each agent can access collective and private rewards, thus enabling a comprehensive understanding of the overall environment dynamics.

There are two types of policy in MARL that reflect the way in which the agents should behave with each other. The first one is the collaboration policy, wherein agents collaborate with each other to accomplish complex tasks. In this case, agents receive information from other agents, enabling efficient teamwork to maximize a collective reward. Conversely, the second policy type is the competition policy, where agents interact with others in a competitive manner. In this case, each agent tries to maximize its own reward while minimizing those of others.

For path-planning problems, a collaborative type of policy should be preferred. Indeed, within the context of aircraft routing at airports, it is imperative to ensure smooth and conflict-free aircraft movements between gates and runways. It is therefore necessary for all aircraft to coordinate their actions in order to avoid potential conflicts and maintain a smooth traffic flow.

3. Methodology

This section presents the methodology for controlling the movements of several aircraft at an airport using reinforcement learning methods. All development and implementation were done using RLlib[2], an open-source Python library developed for reinforcement learning problems. RLlib can be used to train agents in a single-agent or multi-agent environment. Thus, the strategy used in this study was to first train an agent in a single-agent environment, and then to enhance its performance using a multi-agent environment. This strategy enabled the agent to learn individual characteristics, while improving them later in a collaborative environment with other agents. In addition, starting the training process with a pre-trained agent simplifies the complexity associated with the multi-agent problem, accelerates conver-

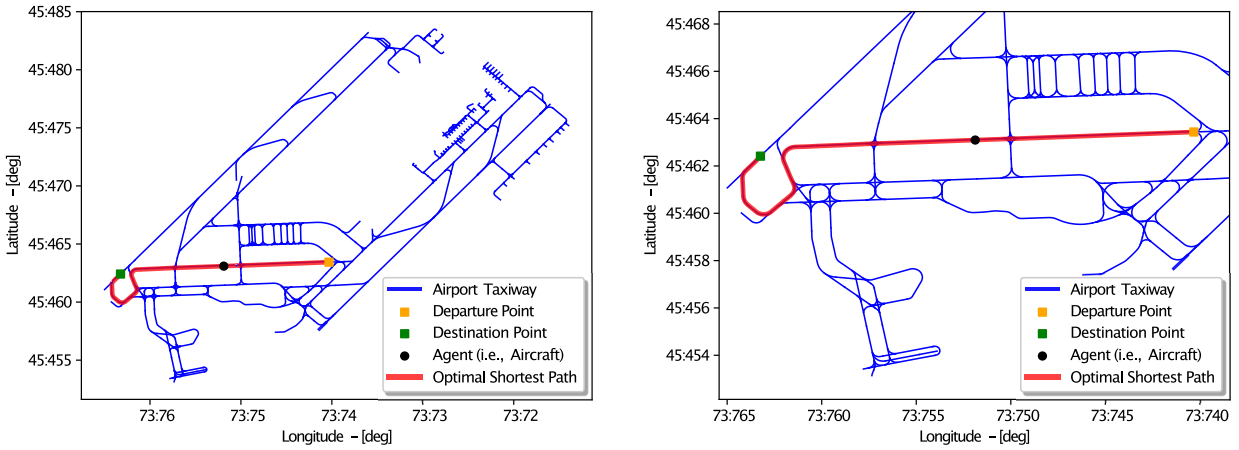
[2] `https://docs.ray.io/en/latest/rllib/index.html`

DOI: 10.13009/EUCASS2023-713

gence, and enables optimal results to be achieved faster.

An agent is an aircraft, while the environment is the airport in which the aircraft must move from a gate (or stand) to a runway (or vice versa). The objective of the agent is, therefore, to learn how to move from a departure node to a destination node by taking the shortest path while complying with displacement constraints. For this purpose, it was assumed that an agent (i.e., an aircraft) can move from one node to another in the graph (i.e., the airport) at each iteration. In addition, at each iteration, the agent receives a set of observations that reflect the current situation and evaluates the best neighboring node to take to reach its destination.

Figure 3 shows an example of a scenario in which an agent follows the optimal shortest path between a departure node and a destination node.



(a) Aircraft Trajectory over the Airport

(b) Zoom around the Aircraft Trajectory

Figure 3: Example of Scenario – The Agent (in black) must reach the Destination Node (in green) from the Departure Node (in orange) by taking the Optimal Path (in red)

3.1 Environment and Observations

The environment is the space in which the agent evolves. In this study, the environment can be any airport represented as an undirected graph. However, for the sake of simplicity, the agent was always trained at the Airport of Montreal (CYUL), which is considered a medium-sized airport. This strategy was used in order to simplify the reinforcement learning problem and to obtain a good agent in an acceptable number of training episodes. However, the agent was tested at other airports, and it was found that, whatever the airport, the agent was able to find the shortest path with a high success rate.

At each iteration k, the agent is assumed to be at a node in the graph, denoted by $P_{[k]}$, and to move to one of the neighboring nodes, denoted by $N^i_{[k]}$ for $i \in \{1, \ldots, N_{[k]}\}$, where $N_{[k]}$ is the number of neighboring nodes. As the number of neighboring nodes can vary according to the agent position in the graph, a maximum number of neighboring nodes was imposed at $N^{\max} = 5$. This number was chosen in order to cover all possible scenarios, even for large airports with multiple possible intersections.

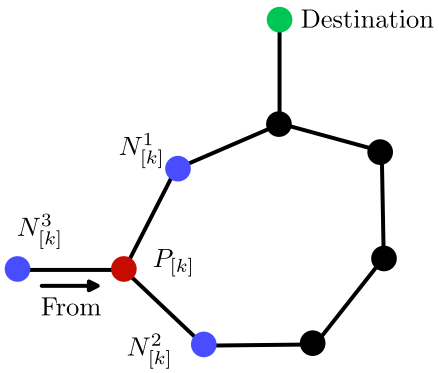
Similarly, at each iteration, the agent is assumed to receive information from the environment in the form of observations. These observations, denoted by $O^i_{[k]}$ for $i \in \{1, \ldots, N^{\max}\}$, correspond to the shortest path distance of each neighboring node to the destination node. It should be noted that taking the shortest path distance instead of the direct distance between the neighboring node and the destination node prevents the agent from being blocked. Indeed, locally, a neighboring node may be the closest to the destination node, however, taking this node could lead the agent to follow a path that will never lead to the destination node. In this case, the agent would be blocked in an area of the airport, and the only possible solution would be to return to a previous node, which is not allowed as aircraft cannot go backward. This aspect can also lead the agent to oscillate between two nodes. However, taking the shortest path distance as an observation ensures that the agent will always be able to reach the destination node.

DOI: 10.13009/EUCASS2023-713

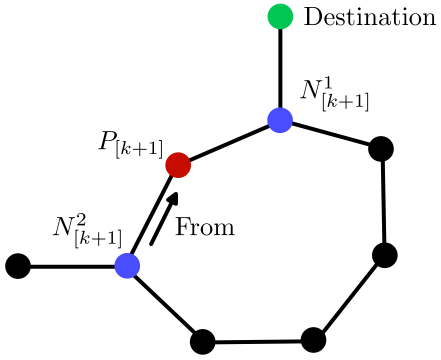
The algorithm used to determine the shortest path between a neighboring node and the destination node was based on a modified A* algorithm. Indeed, it was found that the shortest path calculated by a conventional A* algorithm was not always representative of a feasible trajectory. This algorithm finds the shortest path without taking constraints into account and, in most cases, the optimal solution obtained included turning segments of more than 90°, which is not recommended. To avoid this aspect, a penalty has been added to the calculated path based on the change in the direction between two consecutive segments. The greater the change in direction, the greater the penalty. In addition, another penalty was added, based on the taxi time. As the aircraft ground speed is limited to 10 kts for turning segments (instead of 30 kts), the taxi time increases when the agent uses turns. Thus, by modifying the A* algorithm to take these two penalties into account, the agent was able to avoid sharp turns and limit the number of turns along the path.

Finally, to force the agent to not go backward, a value of -1 is always assigned to the node where the agent comes from. Moreover, if the number of neighboring nodes $N_{[k]}$ is less than $N^{\max}$, then all observation values $O^i_{[k]}$ for $i > N_{[k]}$ are set to -1.

Figure 4 shows an example of observations for two consecutive iterations, k and $k+1$. In this example, each segment has a unit distance, so that the path distance between two nodes is the sum of the segments separating them.



(a) The Agent (in red), the Destination Node (in green), and the Possible Nodes to reach at the Next Iteration (in blue) at k



(b) The Agent (in red), the Destination Node (in green), and the Possible Nodes to reach at the Next Iteration (in blue) at $k+1$

$N^1_{[k]}$	$O^1_{[k]} = 2$
$N^2_{[k]}$	$O^2_{[k]} = 5$
$N^3_{[k]}$	$O^3_{[k]} = -1$
$N^4_{[k]}$	$O^4_{[k]} = -1$
$N^5_{[k]}$	$O^5_{[k]} = -1$

(c) Observations at Iteration k

$N^1_{[k+1]}$	$O^1_{[k+1]} = 1$
$N^2_{[k+1]}$	$O^2_{[k+1]} = -1$
$N^3_{[k+1]}$	$O^3_{[k+1]} = -1$
$N^4_{[k+1]}$	$O^4_{[k+1]} = -1$
$N^5_{[k+1]}$	$O^5_{[k+1]} = -1$

(d) Observations at Iteration $k+1$

Figure 4: Examples of Observations for two Consecutive Iterations

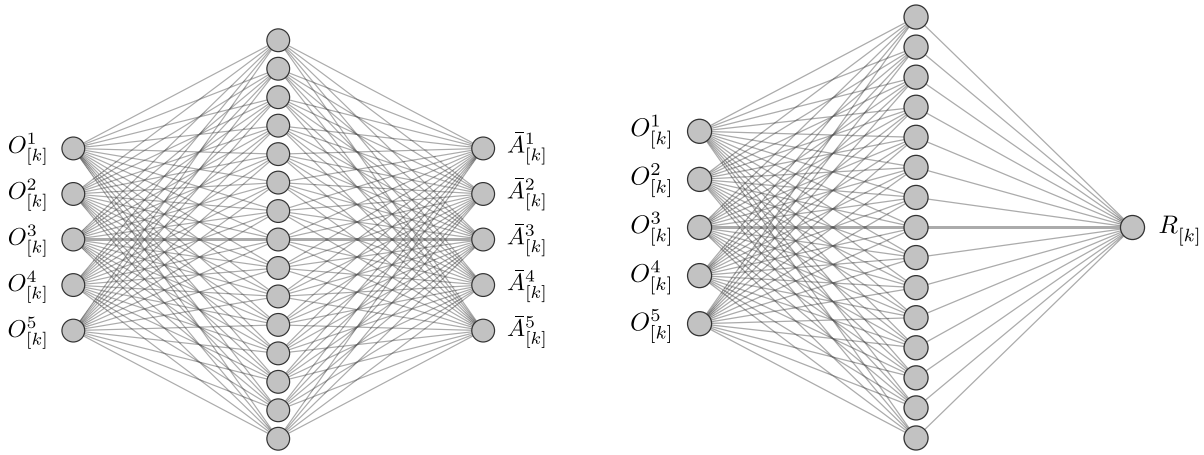
3.2 Agent and Actions

As explained in **Section 2.2**, when using the PPO algorithm, the agent must be characterized by two neural networks: an actor-network and a critic-network. The actor-network is responsible for orienting the action of the agent, while the critic-network is responsible for estimating the reward for a given action. Although actor and critic networks have different objectives, they were assumed to have the same structure. Both networks were modeled by a feedforward neural network with 5 neurons on the input layer, and 15 neurons on the single hidden layer. In addition, a linear activation function was used for the input and output layers, while a ReLU activation function was used for the hidden layer. Figure 5 illustrates the structure of the two neural networks.

The action of the agent represents its choice at a given iteration based on the observations it has received from the environment. For the purposes of this study, it has been assumed that the agent can return through the actor-network

DOI: 10.13009/EUCASS2023-713

5 values, which refer to the probabilities of taking each of the possible neighboring nodes. Thus, for the five possible neighboring nodes, the actor-network will return an output $\bar{A}^i_{[k]}$ for $i \in \{1, \ldots, 5\}$ with $-1 < \bar{A}^i_{[k]} < 1$. The value of -1 means that the node should not be considered a viable solution, while a value of 1 means that the node is the best among all others.



(a) Actor-Network (b) Critic-Network

Figure 5: Illustration of the Agent Neural Networks (i.e., Actor and Critic)

In a complementary way, the critical network receives observations from each neighboring node and outputs the estimated reward.

The PPO then calculates a probability distribution from all the probabilities $\bar{A}_{[k]}$. The action taken by the agent will be chosen using a sample of this distribution. Using a probability instead of the highest probability allows the agent to better explore the environment.

3.3 Stop Conditions and Reward

An episode is composed of several iterations and corresponds to a simulation in which the agent starts moving from a departure node and tries to reach a destination node. Two conditions must be met to end an episode.

The first condition is associated with the agent success. If the agent has reached the destination node within a predetermined maximum number of iterations, then the episode is terminated, and another one is initiated with another random departure and destination nodes.

Conversely, the second condition is associated with the agent failure. To determine this aspect, the number of nodes of the shortest path between the departure and the destination nodes is computed, and denoted as SP_{dist}. If the agent has not reached the destination node after $\alpha \times SP_{\text{dist}}$ iterations, then the episode stops, and another is initiated. The parameter α is called the "*difficulty coefficient*" and decreases over the episodes. This coefficient was introduced to enable better learning, as it adjusts the difficulty during training and then improves the convergences of the neural networks defining the agent. It was set at 10 at the beginning of the training process and gradually decreased to 1 as the agent improved.

In order to encourage the agent to reach the destination node using the minimum number of steps, a negative reward function was used. This function was designed in order to penalize the agent according to its decision (i.e., the node it has selected). Different levels of negative reward were implemented according to the quality of the agent's decision. Table 1 lists the rewards considered in this study, along with their descriptions.

It should be noted that the reward function always has a negative output. Thus, maximizing the reward in this case means obtaining a cumulative reward close to 0.

DOI: 10.13009/EUCASS2023-713

Table 1: Reward Value and Description

Reward	Description
-0.5	The agent has selected the optimal node, defined as being the node leading to the shortest path.
-1.0	The agent has selected the sub-optimal node, characterized as a viable node that leads to the destination node but does not belong to the shortest path.
-1.5	The agent has selected the worst node among the sub-optimal nodes leading to the destination node.
-5.0	The agent has selected an invalid node (i.e., with an observation of -1).

4. Results and Validation

This section presents the results obtained for the proposed methodology. The agent was trained over 600,000 iterations. In addition, the hyperparameters required to train the actor and critic networks, such as the learning rate, batch size, and weights update frequency, were chosen according to the values given in Table 2.

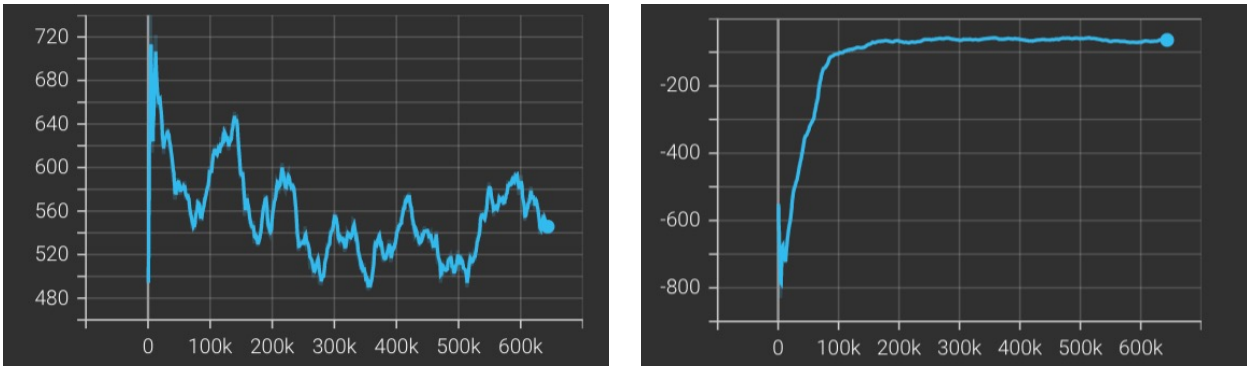
Table 2: Hyperparameters used for the Training of the Agent

Reward	Description	
Lr_{critic}	4×10^{-5}	Learning rate for the critic-network
Lr_{actor}	4×10^{-5}	Learning rate for the actor-network
$BatchSize$	6000	Number of iterations for one update
τ	12,000	Number of iterations between updates

The values shown in Table 2 were obtained by trial and error, in order to find the best compromise between calculation time and results efficiency.

4.1 Evolution of the Reward

Figure 6 shows the average number of iterations per episode (Fig. 6a) and the total reward (Fig. 6b) as a function of the total number of training iterations.



(a) Number of Iterations per Episode as a Function of the Total Number of Training Iterations

(b) Total Reward per Episode as a Function of the Total Number of Training Iterations

Figure 6: Evolution of Average Episode Length and Reward over the Training Process

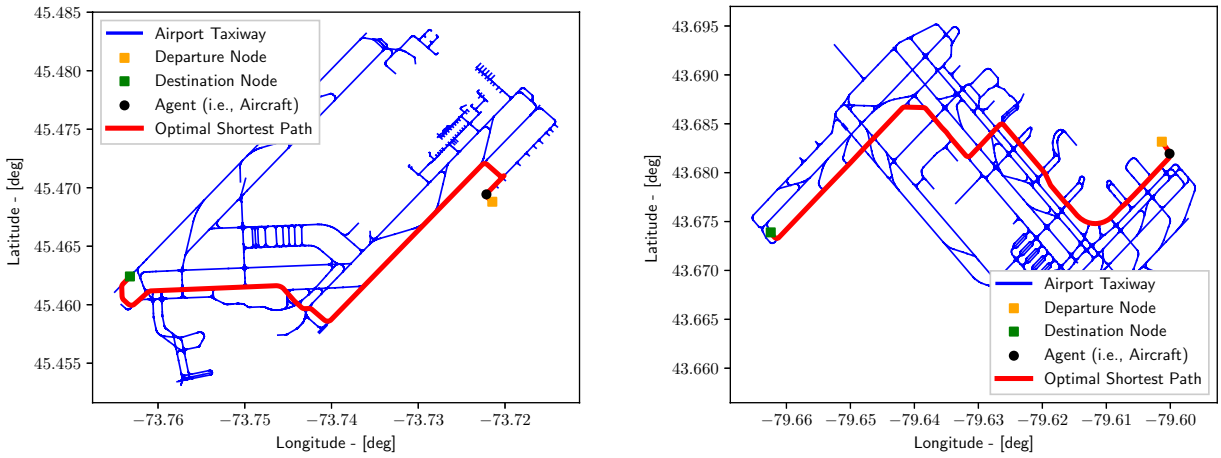
As shown in Fig. 6a the general trend of the curve indicates that the number of iterations per episode decreases with the number of training iterations. This trend indicates that, as training progresses, the agent became increasingly efficient, requiring fewer iterations to complete an episode. In other words, the agent was able to reach the destination node with fewer iterations by following the shortest path. Regarding the result in Fig. 6b, it is interesting to note that the total reward per episode increased logarithmically to reach a constant value of around -80. It is worth noticing that the reward is always negative. Therefore, during the training process, the PPO tried to maximize the reward by approaching the limit value of 0. The maximum reward value was obtained after 150,000 (150 K) training iterations. For a typical

DOI: 10.13009/EUCASS2023-713

reinforcement learning problem, this number of iterations is relatively low, demonstrating that the strategy of using an airport as an environment with a single-agent approach allows to obtain a pre-trained agent very rapidly.

4.2 Validation for Different Scenarios

Once the agent trained over the airport of Montreal, an analysis was conducted in order to evaluate it reliability and robustness. For this purpose, multiple scenarios were done by varying the departure node and arrival node, as well as the airport. Figure 7 shows two examples of scenario at Montreal Airport (Fig. 7a) and Toronto Airport (Fig. 7b).



(a) Aircraft Trajectory at the Montreal Airport (CYUL)

(b) Aircraft Trajectory at the Toronto Airport (CYYZ)

Figure 7: Examples of Validation Scenarios for Two Different Airports

In order to determine if the agent was successful or if it failed in finding the optimal path, a comparison was done between the path estimated by the agent and the one determined using the modified A* algorithm and based on the following success coefficient:

$$\delta = \frac{N_{\text{optimal}}}{N_{\text{agent}}} \times 100 \tag{1}$$

where N_{optimal} is the number of nodes composing the optimal path determined using the modified A* algorithm, and N_{agent} is the number of nodes composing the path found by the agent. As the modified A* algorithm always finds the optimal path, the following inequality is always satisfied: $N_{\text{optimal}} \leq N_{\text{agent}}$. Consequently, the coefficient δ can never be greater than 100%, and this value means that the agent has found the same optimal path as the modified A* algorithm. Conversely, when δ is lower than 100% this means that the agent has found a sub-optimal solution.

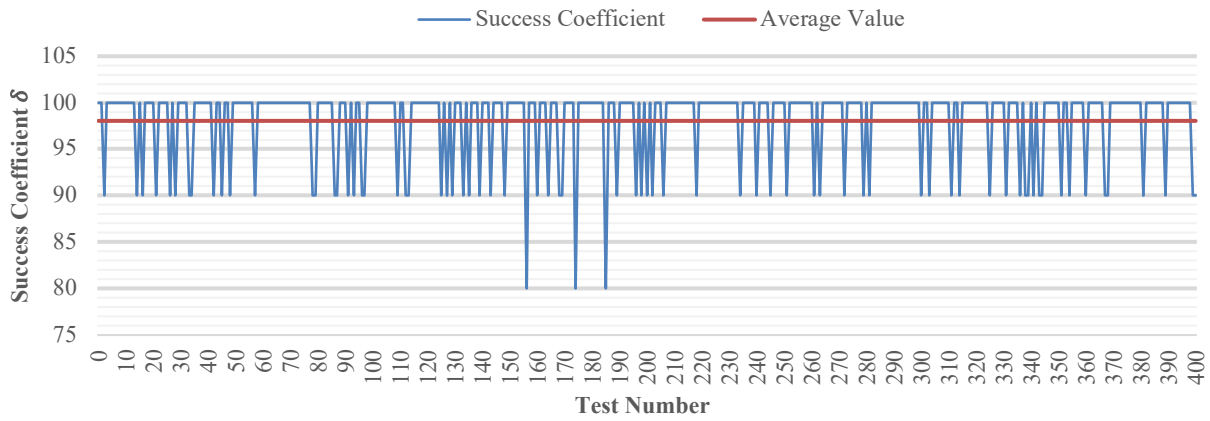


Figure 8: Variation of the Success Coefficient for 400 Scenarios

DOI: 10.13009/EUCASS2023-713

Figure 8 shows the variation in the success coefficient over 400 test scenarios. As it can be seen in this figure, the success coefficient varied mainly between 90% and 100%, with only three scenarios where the coefficient was equal to 80%. It was found that the mean value of the success coefficient was around 98%, with a standard deviation of 4.15%.

It should be noted that for the scenarios where the success coefficient was less than 100%, meaning that the agent has failed to find the shortest path, the path length determined by the agent has remained very close to its optimal path length. Consequently, even if the solution found by the agent was sub-optimal, it can be considered an acceptable solution, as the difference in terms of length cost was negligible.

4.3 Validation for a Multi-Agent Environment

Once the agent has been validated in a single-agent environment, it was next trained into a multi-agent environment with other agents (i.e., aircraft). It should be noted that, as a first study, all agents were trained to move within the environment in order to reach their destination nodes using the shortest path. Since the ground speed was not considered a possible action, conflict management between two agents has not been taken into account. Nevertheless, this aspect will be added in a future study in order to improve the agent policy for adjusting ground speed to avoid a potential conflict.

Figure 9 shows an example of simulation results obtained for 5 agents (i.e., aircraft) at the Montreal airport. In this figure, each agent path is represented by a different color, with departure nodes represented by an orange square and destination nodes by a green square. All agents had the same policy, meaning that they were all defined using the same actor- and critic-network.

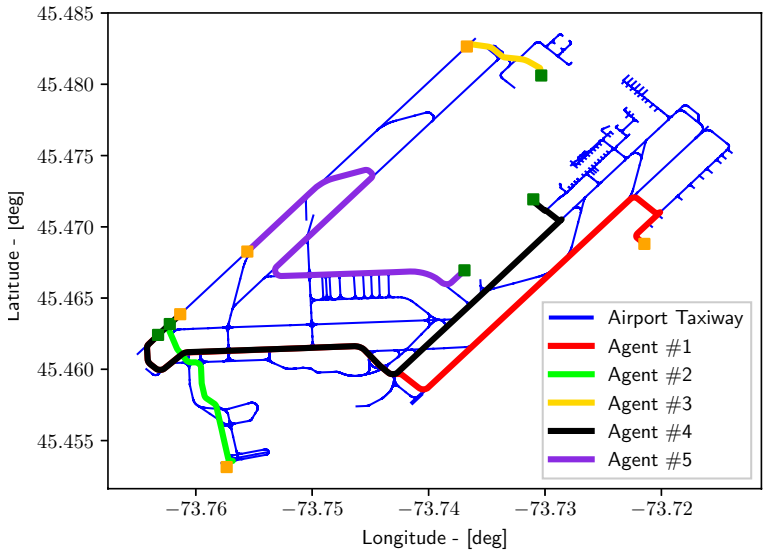


Figure 9: Validation Scenario for Multi-Agent (i.e, Multi-Aircraft) at the Montreal Airport

Overall, it is interesting to note that all agents were able to reach their destination nodes by following an optimal path that corresponds to the shortest path without taking sharp turns, and avoiding turn segments as much as possible. It should be noted that when an agent arrives at its destination node before the others, it does not move and remains in this position, waiting for the other agents to reach their destination nodes. It is also interesting to notice that Agent #1 (in red) and Agent #2 (in black) followed the same trajectory when they arrived at the same node. This aspect is very important, as it demonstrates the consistency of the agents in providing the same optimal path from the same departure node.

The results obtained in this section are therefore very good, as they demonstrate that using a multi-agent reinforcement learning approach can be a very good solution for controlling aircraft movements and improving aircraft routing at airports.

DOI: 10.13009/EUCASS2023-713

5. Conclusion

This paper presents an innovative methodology for managing aircraft ground trajectories at airports using reinforcement learning methods. An agent is represented by an aircraft, while the environment is assumed to be an airport modeled as an undirected graph. The agent was first trained in a single-agent environment, assuming that only one aircraft could move in the airport. Then, it was trained using the PPO (Proximal Policy Optimization) algorithm, which is a model-free gradient learning algorithm that teaches the agent to take the optimal path between a departure node and a destination node. The optimal path was defined as the shortest path with the fewest turning segments by avoiding all non-recommended turns (i.e., turns with a change of direction greater than 90°). Results have shown that after 100,000 training episodes, the agent was able to find the optimal path with an average success rate of 98%.

The agent was then tested in a multi-agent environment, considering several aircraft moving in the airport. Here again, it was found that the agents were able to move freely around the airport and reach their destination nodes using the optimal path. These results are very interesting, as they demonstrate that an agent that has been trained in a single-agent environment can also perform very well in a multi-agent environment.

In this study, the agent policy did not take conflict management into account. It would therefore be interesting to extend the study by adding speed management to the list of actions that the agent can undertake. This would make it possible to adjust the agent speed or, in extreme cases, to stop it in order to avoid a potential conflict with other agents. The addition of speed control as an action would also make it possible to take into account constraints related to the time of arrival at a destination node, in order to better plan aircraft movements on the ground.

References

[1] International Air Transport Association (IATA), "IATA Forecast Predicts 8.2 billion Air Travelers in 2037," , 2020. [Online] https://www.iata.org/en/pressroom/pr/2018-10-24-02/.

[2] International Air Transport Association (IATA), "International Air Transport Association Annual Review 2022," , 2022. [Online] https://www.iata.org/en/publications/annual-review/.

[3] Owen, B., Anet, J. G., Bertier, N., Christie, S., Cremaschi, M., Dellaert, S., Edebeli, J., Janicke, U., Kuenen, J., Lim, L., and Terrenoire, E., "Review: Particulate Matter Emissions from Aircraft," *Atmosphere*, Vol. 13, No. 8, 2022, p. 1230. https://doi.org/10.3390/atmos13081230.

[4] Quadros, F. D. A., Snellen, M., and Dedoussi, I. C., "Regional Sensitivities of Air Quality and Human Health Impacts to Aviation Emissions," *Environmental Research Letters*, Vol. 15, No. 10, 2020, p. 105013. https://doi.org/10.1088/1748-9326/abb2c5.

[5] Ma, L., Tian, Y., Zhang, Y., and Chu, P., "Trajectory Optimization of Aircraft for A Continuous Descent Continuous Procedure," IEEE, Shanghai, China, 2020, pp. 2063–2067. https://doi.org/10.1109/CAC51589.2020.9326515.

[6] Alharbi, E. A., Abdel-Malek, L. L., Milne, R. J., and Wali, A. M., "Analytical Model for Enhancing the Adoptability of Continuous Descent Approach at Airports," *Applied Sciences*, Vol. 12, No. 3, 2022, p. 1506. https://doi.org/10.3390/app12031506.

[7] Liv, A., Dancila, R., and Botez, R. M., "Trajectory Optimization Algorithm for a Constant Altitude Cruise Flight with a Required Time of Arrival Constraint," American Institute of Aeronautics and Astronautics, Dallas, TX, 2015. https://doi.org/10.2514/6.2015-2282.

[8] Jensen, L., Hansman, R. J., Venuti, J. C., and Reynolds, T., "Commercial Airline Speed Optimization Strategies for Reduced Cruise Fuel Consumption," American Institute of Aeronautics and Astronautics, Los Angeles, CA, USA, 2013. https://doi.org/10.2514/6.2013-4289.

[9] Murrieta-Mendoza, A., Botez, R. M., and Félix Patrón, R. S., "Flight Altitude Optimization Using Genetic Algorithms Considering Climb and Descent Costs in Cruise with Flight Plan Information," 2015, pp. 2015–01–2542. https://doi.org/10.4271/2015-01-2542.

[10] Jensen, L., Hansman, R. J., Venuti, J., and Reynolds, T., "Commercial Airline Altitude Optimization Strategies for Reduced Cruise Fuel Consumption," American Institute of Aeronautics and Astronautics, Atlanta, GA, USA, 2014. https://doi.org/10.2514/6.2014-3006.

DOI: 10.13009/EUCASS2023-713

[11] Ghazi, G., and Botez, R. M., "Aircraft Mathematical Model Identification for Flight Trajectories and Performance Analysis in Cruise," *Journal of Aerospace Information Systems*, Vol. 19, No. 8, 2022, pp. 530–549. https://doi.org/10.2514/1.I011050.

[12] Ghazi, G., Botez, R. M., and Domanti, S., "New Methodology for Aircraft Performance Model Identification for Flight Management System Applications," *Journal of Aerospace Information Systems*, Vol. 17, No. 6, 2020, pp. 294–310. https://doi.org/10.2514/1.I010791.

[13] Ghazi, G., Kossinga-Yalemba, M., and Botez, R. M., "Methodology to Identify a Mathematical Model for Predicting Cessna Citation X Cruise Performance in Cruise Regime using Flight Manual Data," EUCASS association, Madrid, Spain, 2019. https://doi.org/10.13009/EUCASS2019-496.

[14] Dancila, R., and Botez, R., "Vertical Flight Profile Optimization for a Cruise Segment with RTA Constraints," *The Aeronautical Journal*, Vol. 123, No. 1265, 2019, pp. 970–992. https://doi.org/10.1017/aer.2019.47.

[15] Murrieta-Mendoza, A., and Botez, R., "Lateral Navigation Optimization Considering Winds and Temperatures for Fixed Altitude Cruise Using Dijsktra's Algorithm," American Society of Mechanical Engineers, Montreal, Quebec, Canada, 2014. https://doi.org/10.1115/IMECE2014-37570.

[16] Murrieta-Mendoza, A., Romain, C., and Botez, R. M., "Commercial Aircraft Lateral Flight Reference Trajectory Optimization," *IFAC-PapersOnLine*, Vol. 49, No. 17, 2016, pp. 1–6. https://doi.org/10.1016/j.ifacol.2016.09.001.

[17] Murrieta-Mendoza, A., Romain, C., and Botez, R. M., "3D Cruise Trajectory Optimization Inspired by a Shortest Path Algorithm," *Aerospace*, Vol. 7, No. 7, 2020, p. 99. https://doi.org/10.3390/aerospace7070099.

[18] Murrieta-Mendoza, A., Beuze, B., Ternisien, L., and Botez, R. M., "New Reference Trajectory Optimization Algorithm for a Flight Management System Inspired in Beam Search," *Chinese Journal of Aeronautics*, Vol. 30, No. 4, 2017, pp. 1459–1472. https://doi.org/10.1016/j.cja.2017.06.006.

[19] Murrieta-Mendoza, A., Botez, R. M., and Bunel, A., "Four-Dimensional Aircraft en Route Optimization Algorithm using the Artificial Bee Colony," *Journal of Aerospace Information Systems*, Vol. 15, No. 6, 2018, pp. 307–334. https://doi.org/10.2514/1.I010523.

[20] Guclu, O. E., and Cetek, C., "Analysis of Aircraft Ground Traffic Flow and Gate Utilisation using a Hybrid Dynamic Gate and Taxiway Assignment Algorithm," *The Aeronautical Journal*, Vol. 121, No. 1240, 2017, pp. 721–745. https://doi.org/10.1017/aer.2017.20.

[21] Yu, C., Zhang, D., and Henry Lau, H., "A Heuristic Approach for Solving an Integrated Gate Reassignment and Taxi Scheduling Problem," *Journal of Air Transport Management*, Vol. 62, 2017, pp. 189–196. https://doi.org/10.1016/j.jairtraman.2017.04.006.

[22] Guépet, J., Briant, O., Gayon, J.-P., and Acuna-Agost, R., "Integration of Aircraft Ground Movements and Runway Operations," *Transportation Research Part E: Logistics and Transportation Review*, Vol. 104, 2017, pp. 131–149. https://doi.org/10.1016/j.tre.2017.05.002.

[23] Benlic, U., Brownlee, A. E., and Burke, E. K., "Heuristic Search for the Coupled Runway Sequencing and Taxiway Routing Problem," *Transportation Research Part C: Emerging Technologies*, Vol. 71, 2016, pp. 333–355. https://doi.org/10.1016/j.trc.2016.08.004.

[24] Gotteland, J. B., and Durand, N., "Genetic algorithms Applied to Airport Ground Traffic Optimization," IEEE, Canberra, ACT, Australia, 2003, pp. 544–551. https://doi.org/10.1109/CEC.2003.1299623.

[25] Brownlee, A. E. I., Woodward, J. R., Weiszer, M., and Chen, J., "A Rolling Window with Genetic Algorithm Approach to Sorting Aircraft for Automated Taxi Routing," ACM, Kyoto Japan, 2018, pp. 1207–1213. https://doi.org/10.1145/3205455.3205558.

[26] Dabachine, Y., Bouikhalene, B., and Balouki, A., "Bidirectional Search Algorithm for Airport Ground Movement," IEEE, Werdanye, Lebanon, 2018, pp. 1–9. https://doi.org/10.1109/ACIT.2018.8672668.

[27] Lesire, C., "An Iterative A* Algorithm for Planning of Airport Ground Movements," *Frontiers in Artificial Intelligence and Applications : ECAI 2010*, Vol. 215, 2010, pp. 413 – 418. https://doi.org/10.3233/978-1-60750-606-5-413.

DOI: 10.13009/EUCASS2023-713

[28] Zhou, H., and Jiang, X., "Research on Taxiway Path Optimization Based on Conflict Detection," *PLOS ONE*, Vol. 10, No. 7, 2015, p. e0134522. https://doi.org/10.1371/journal.pone.0134522.

[29] Ravizza, S., Atkin, J. A. D., and Burke, E. K., "A More Realistic Approach for Airport Ground Movement Optimisation with Stand Holding," *Journal of Scheduling*, Vol. 17, No. 5, 2014, pp. 507–520. https://doi.org/10.1007/s10951-013-0323-3.

[30] Szymanski, M., Ghazi, G., and Botez, R. M., "Development of a Map-Matching Algorithm for the Analysis of Aircraft Ground Trajectories using ADS-B Data," American Institute of Aeronautics and Astronautics, San Diego, CA and Online, 2023. https://doi.org/10.2514/6.2023-3758.

[31] Ding, Z., Huang, Y., Yuan, H., and Dong, H., "Introduction to Reinforcement Learning," *Deep Reinforcement Learning: Fundamentals, Research and Applications*, 2020, pp. 47–123.

[32] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O., "Proximal Policy Optimization Algorithms," 2017. https://doi.org/10.48550/ARXIV.1707.06347.