Thirteenth USA/Europe Air Traffic Management Research and Development Seminar (ATM2019)

A Machine Learning Approach for Conflict Resolution in Dense Traffic Scenarios with Uncertainties

Duc-Thinh Pham, Ngoc Phu Tran, Sameer Alam and Vu Duong
Air Traffic Management Research Institute,
School of Mechanical & Aerospace Engineering,
Nanyang Technological University, Singapore

Daniel Delahaye
OPTIM Research Lab
Ecole Nationale de l'Aviation Civile
Toulouse, France

***Abstract*—With the continuous growth in the air transportation demand, air traffic controllers will have to handle increased traffic and consequently more potential conflicts. That gives rise to the need for conflict resolution tools that can perform well in high-density traffic scenarios given a noisy environment. Unlike model-based approaches, learning-based or machine learning approaches can take advantage of historical traffic data and flexibly encapsulate the environmental uncertainty. In this study, we propose an artificial intelligent agent that is capable of resolving conflicts, in the presence of traffic and given uncertainties in conflict resolution maneuvers, without the need of prior knowledge about a set of rules mapping from conflict scenarios to expected actions. The conflict resolution task is formulated as a decision-making problem in large and complex action space, which is applicable for employing the reinforcement learning algorithm. Our work includes the development of a learning environment, scenario state representation, reward function, and learning algorithm. As a result, the proposed method, inspired from Deep Q-learning and Deep Deterministic Policy Gradient algorithms, can resolve conflicts, with a success rate of over 81%, in the presence of traffic and varying degrees of uncertainties.**

***Index Terms*—reinforcement learning, air traffic control, deep deterministic policy gradient, conflict resolution**

I. Introduction

Air traffic control (ATC) plays a crucial role in the air traffic management (ATM) system as it is responsible for maintaining flights safety and efficiency. Air Traffic Controllers (ATCOs) must maintain a safe separation distance between any two aircraft at all times. Conflict or loss of separation, between any two aircraft, occurs when the distance between them is smaller than the separation standard, for example, 5 nautical miles laterally and 1000 feet vertically during the en-route phase of flight. When a potential loss of separation is detected, ATCOs are responsible for issuing resolution advisory, or vector instructions, to one or both aircraft to resolve the conflict. A maneuver may include a heading change or speed change for lateral conflict resolutions, or a flight level change (climb or descend) for vertical conflict resolutions. With the continuous growth in the air transportation demand [1], ATCOs will have to deal with increased traffic in their respective sectors. In such a situation, conflict resolution tools to needed to support ATCOs in high-density traffic scenarios, with uncertainties in the environment.

Many mathematical models for conflict resolution have been proposed in the literature. For a comprehensive review see Yang et al. [2]. Some recent works look into enhancing the capability of such automated conflict solvers. For instances, Yang et al. [3] used probability reach sets to represent aircraft locations, and aircraft deconfliction is performed by separating these reach sets using second-order cone programming with aircraft dynamics considered. However, this approach does not perform well in handling a large number of aircraft with uncertainty. In the recent research, Hao et al. [4] employed aircraft reachable space, where conflict resolution scheme accounts for the intent of the aircraft via aircraft's space-time Prism. The execution time for this method scales up significantly with the number of aircraft involved, especially when a fine grid is applied. Model predictive control (MPC) is also a promising approach to conflict resolution. Yokohama [5] applied MPC to perform trajectory prediction and conflict resolution simultaneously, in which the aircraft separation condition is implicitly imposed during trajectory prediction. However, the mathematical model is highly complex, and the resolution quality depends on the quality (noise free) of available historical data. MPC was also employed in the work of Jikov et al. [6], in which the authors proposed multiple models for conflict resolution considering the minimization of the cost due to the maneuver, using the efficient algorithm. In another approach, advanced surrounding traffic analysis was proposed as the basis for conflict resolution decision [7]. The analysis of surrounding traffic includes the concept of aerial ecosystem and traffic complexity evaluation for the determination of resolution, in which the domino effect, i.e., the number of aircraft causally involved in the separation service, is considered. Large scale conflict resolution models were also proposed by Allignol et al. [8] and Liu et al. [9]. While the work in [9] uses aircraft location network and limits its resolution's maneuver to velocity adjustment only, the model provided in [8] provides a for 3D conflict resolution with limited uncertainty. From our observation of

the literature, mathematical models for conflict resolution have several common limitations. First, complete knowledge of the mapping from conflict scenarios to maneuvers is required; this makes mathematical models highly complex and results in poor quality resolutions in the presence of high uncertainty, as the full knowledge about the environmental uncertainty could never be obtained. Second, the input scenarios must be well standardized for the mathematical models to work properly, and the models do not self-evolve when dealing with unseen and non-standard scenarios. In this work, we attempt to overcome these drawbacks by considering machine learning approach for conflict resolution, as learning method does not require prior knowledge of how to efficiently resolve a conflict, and learning algorithm can self-evolve when being exposed to unseen scenarios.

Machine learning methods have emerged as promising method for solving air traffic management problems such as Taxi-out time prediction [10], [11], aircraft sequencing [12], trajectory prediction [13], aircraft performance parameter predicting [14], air traffic flow extraction [15], flight delay prediction [16], [17]. Deep learning models, like Long Short-Term Memory (LSTM), are also investigated in [18] for air traffic delay prediction tasks.

For decision-making problems like conflict resolution, their large and continuous state and action spaces are a challenge for machine learning methods. However, Reinforcement Learning (RL) can be considered as one of the promising approaches for their success in building the playing engine for classic board game with expert levels [19] such as in Backgammon, Checker and Scrabble. Moreover, advanced machine learning algorithms, like deep learning, have demonstrated breakthroughs such as Deep Blue [20] for Chess, Poker-CNN [21] and DeepStack [22] for Poker. Recently, the combination of deep learning and reinforcement learning which is called deep reinforcement learning (DRL) has increased the potential of automation for many decision-making problems that were previously intractable because of their high-dimensional state and action spaces. In 2015, Mnih et al. [23] introduced the Deep Q-Network model which could learn to play a range of Atari 2600 video games at a superhuman level, directly from raw image pixels. Secondly, AlphaGo [24], that defeated a human world champion in Go used neural networks that were trained using supervised and RL, in combination with a traditional heuristic search algorithm. Differentiating from those works dealing with discrete action space, [25], [26] has tackled the problem of continuous action space by introducing a general-purpose continuous DRL framework, the actor-critic Deterministic Policy Gradient Algorithms. The action policy function is approximated by a neural network while the reward function estimator is trained with the second one.

In this study, we develop an AI agent that is capable of resolving conflicts in the presence of surrounding traffic and uncertainty. The AI agent resolves conflict in lateral dimension only. A RL algorithm is developed as a learning model which can handle the large and continuous state and action spaces of conflict scenario. Like ATCOs, the AI agent can also learn

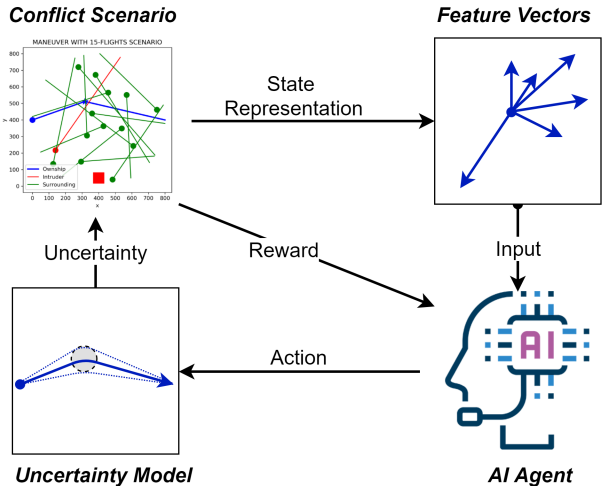

Figure 1. Interaction between the learning environment and the AI agent

and form its strategy when dealing with conflict scenarios and have the ability to self-evolve via trial-and-error.

The process of training the AI agent for conflict resolution is illustrated in Figure 1). Conflict scenarios involving multiple aircraft are generated and presented to the AI agent by the learning environment. The agent, which is driven by RL algorithm, learns to resolve these conflicts by applying several maneuvers given the environmental uncertainty. The agent receives a reward for every maneuver it has tried as performance feedback, and the value of the reward depends on the quality of the maneuvers: positive rewards for maneuvers that successfully resolve the conflicts and negative rewards, or penalties, for maneuvers that are unable to separate the conflicting aircraft safely. The learning objective is to maximize the reward, and the agent is assumed trained when it consistently gains (converges) high rewards for resolving unseen conflict scenarios.

Our main contribution is formulation of the conflict resolution problem as a decision-making problem which is suitable for the reinforcement learning algorithm. To accomplish this, we give special considerations to the following sub-tasks.

1) Unlike the time-based continuous control problem reported in [26], our problem is designed as a space-based searching action where an agent will perform a list of actions at a given time.
2) Developing a learning environment for flight conflict detection and resolution that possesses the following characteristics.
   - The reward function is carefully designed to consider not only the conflict status of the scenario but also the quality (e.g., deviation, maneuverability, etc.) of the maneuvers.
   - Maneuver's uncertainty is encapsulated in a learning model with different level.
   - A novel scenario representation (state vector) is proposed which contains information of conflict scenario such as conflict status, optimal status and

uncertainty level. This state vector must be carefully designed in order to guarantee the convergence of the training.

3) The learning model is designed to handle multi-dimensional actions with different physical scales and units (e.g. time and distance).

II. Learning Environment

In the RL for conflict resolution (Figure 1), the main roles of the learning environment are to (1) present its state to the agent in a form that provides sufficient information to support the agent's decision-making, (2) receive and evaluate the agent's action, and (3) give feedback to the agent as a reward. To provide such environment to the agent for learning to resolve flight conflict, we develop a scenario generator that generates conflict scenarios and represents them in a form perceivable to the agent. Also, the agent's action is defined and the mapping from the agent's actions to the maneuvers taken by the ownship is established. A reward function is designed for the assessment of the maneuver suggested by the agent. We also consider the environmental uncertainty that occurs during the implementation of the agent's actions in order to assess its learning performance.

A. Conflict scenarios

We define a conflict scenario as a traffic scenario that occurs within a circular area of interest (airspace) of radius r, in which there is one pair of potential conflict between an ownship and an intruder aircraft, in the presence of surrounding traffic. An example of a conflict scenario considered in this study is shown in Figure 2a, and the conflict pair between the ownship and the intruder in this scenario is separately plotted in Figure 2b for clear presentation. We assume no conflict among the surrounding aircraft; in other words, a conflict always occurs between the ownship and another aircraft in the given airspace. Here, for convenience and without loss of generality, we generate conflict scenarios such that the ownships always point along the horizontal direction and the traveling distance of the ownship in the given airspace is equal to the diameter of the airspace's boundary. Any direction of the ownship could be achieved by performing a simple rotational transformation of the interested airspace, and different travelling distances of the ownship could be considered by simply setting the size of the interested area.

Let n be the number of aircraft in the given airspace when a potential conflict is being considered, $\mathbf{A}_i$ denote the locations of the aircraft at the moment the conflict scenario presented to the agent ($t_0 = 0$), and $\mathbf{B}_i$ the locations where the aircraft exit the given airspace ($0 \leq i < n$), see Figure 2a. Consequently, $\mathbf{A}_i\mathbf{B}_i$ represent the aircraft's trajectories and $\overrightarrow{\mathbf{A}_i\mathbf{B}_i}$ are the initial headings of the aircraft. If the aircraft continue their journeys with this original flights plan, the ownship (following route $\mathbf{A}_0\mathbf{B}_0$) and the intruder (following route $\mathbf{A}_1\mathbf{B}_1$) are converging; they will simultaneously reach $\mathbf{P}$ and $\mathbf{Q}$ (Figure 2b). Since the scenario is generated such that the distance $\mathbf{PQ}$ is less than d_{sep}, which is the safe separation to maintain, the

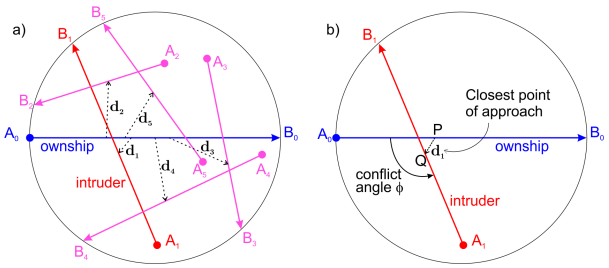

Figure 2. a) Example of a conflict scenario involving a two-aircraft conflict in the presence of four surrounding aircraft. $\mathbf{A}_0\mathbf{B}_0$ is the ownship and $\mathbf{A}_1\mathbf{B}_1$ is the intruder. b) The conflict pair where $\mathbf{PQ}$ is the closest distance between two aircraft.

two aircraft are losing their safe separation if none of them takes any maneuver. Here, $\mathbf{PQ}$ is called the closest point of approach (CPA) between the ownship and the intruder, also denoted by the CPA closure vector $\overrightarrow{d_1}$ from $\mathbf{P}$ to $\mathbf{Q}$. Similarly, the CPAs between the ownship and the surrounding aircraft are denoted by $\overrightarrow{d_i}$ where $2 \leq i < n$. Note that at the beginning, $\|\overrightarrow{d_1}\| < d_{\text{sep}}$ while $\|\overrightarrow{d_i}\| \geq d_{\text{sep}}$, $2 \leq i < n$; this imposes the single initial conflict condition to the generated scenarios, which is the interest of this work.

We now briefly describe the computation of CPA between the ownship and the intruder, and the same procedure is applied to find CPA between the ownship and the surrounding aircraft. Assume that all aircraft are cruising at the same speed of v_c. At $t_0 = 0$, the ownship is at $\mathbf{A}_0$ and the intruder $\mathbf{A}_1$. The velocities of the ownship and the intruder are $\vec{u} = v_c(\overrightarrow{\mathbf{A}_0\mathbf{B}_0}/\|\overrightarrow{\mathbf{A}_0\mathbf{B}_0}\|)$ and $\vec{v} = v_c(\overrightarrow{\mathbf{A}_1\mathbf{B}_1}/\|\overrightarrow{\mathbf{A}_1\mathbf{B}_1}\|)$, respectively. At a time $t > 0$, the locations of the the ownship and the intruder are respectively given by $\overrightarrow{P}(t) = \overrightarrow{\mathbf{A}_0} + \vec{u}t$ and $\overrightarrow{Q}(t) = \overrightarrow{\mathbf{A}_1} + \vec{v}t$, and distance between them renders as $d_1(t) \equiv \|\overrightarrow{d_1}\| = \overrightarrow{W_0} + (\vec{u} - \vec{v})t$ where $\overrightarrow{W_0} = \overrightarrow{\mathbf{A}_0\mathbf{A}_1}$. Minimizing $d_1(t)$ yields the time to CPA as $t_{\text{CPA}} = -\overrightarrow{W_0} \cdot (\vec{u} - \vec{v})/\|\vec{u} - \vec{v}\|^2$, and the closure at CPA as $d_{1(\text{CPA})} = d_1(t_{\text{CPA}})$.

B. Ownship's Maneuver

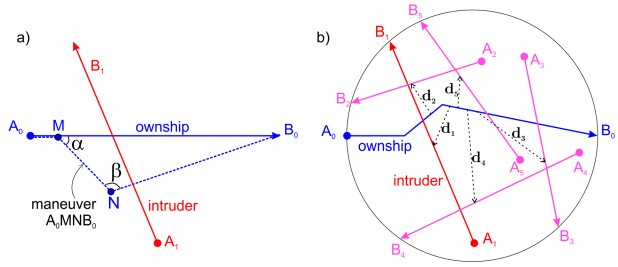

Figure 3. a) An example of maneuver. The ownship makes a heading change α° at point $\mathbf{M}$ at $t = t_1$, continues in the new heading $\mathbf{MN}$ during t_2 seconds, and heading back towards original end point at return point $\mathbf{N}$. A maneuver is fully defined by a set of three parameters (t_1, α, t_2). b) The maneuver implemented in the traffic scenario.

A maneuver, e.g. maneuver $\mathbf{A}_0\mathbf{MNB}_0$ in Figure 3a, is defined as a series of actions performed by the ownship:

deviate from original path at time t_1 seconds and at location $\mathbf{M}$ (measuring from $t_0 = 0$ at $\mathbf{A}_0$) by changing the heading by an angle α, and then keep heading along vector $\overrightarrow{\mathbf{MN}}$ in t_2 seconds before heading back towards $\mathbf{B}_0$ at return point $\mathbf{N}$. Thus, a maneuver is fully defined by a set of three parameters (t_1, α, t_2). In addition, a valid maneuver is defined as the maneuver that satisfies $t_1 < t_{\text{CPA}}$, $\|\alpha\| < 90$ degrees, and t_2 takes a value such that the return point $\mathbf{N}$ located within the interested area. In this study, we assume that any applied maneuver modifies the path of the ownship while leaves the intruder's path unchanged.

Figure 3b demonstrates an example of a maneuver being implemented in a scenario. An employed maneuver changes the scenario from the current state into the next one by updating the CPA closure vectors $\overrightarrow{d_i}$. The quality of a maneuver, therefore, is essentially determined by these CPA closure vectors, which reflect the aircraft's separation status in the scenario. We shall elaborate the evaluation of the agent's actions and the resultant maneuvers in the definition of reward function later in this section.

C. Environmental uncertainty

The working environment in air traffic control have high degrees of uncertainties. The controllers have to deal with unknowns originated from, for example, inaccurate trajectory prediction, equipment's measurement errors, weather, and other unexpected events in the airspace. Therefore, any conflict resolution tool for ATCO must perform effectively in the presence of uncertainty. In this work, we consider environmental uncertainty as something that affects the accurate/precise implementation of the agent's conflict resolution actions.

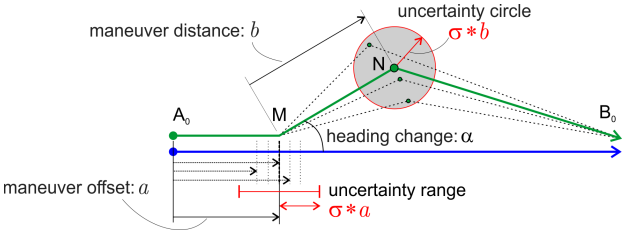

Figure 4. Environmental uncertainty and its impact on the agent's action

Consider a situation in which the agent suggests a maneuver determined by (t_1, α, t_2), e.g the green maneuver as shown in Figure 4, where a is the expected maneuver offset, α the expected heading change, and b the expected maneuver distance. The ownship is expected to change its heading at the expected heading change point $\mathbf{M}$ and turn back at the expected return point $\mathbf{N}$. Due to environmental uncertainties, however, the actual maneuver is slightly deviated from the expected one. In particular, the actual heading change point $\mathbf{M}'$ is determined by the actual maneuver offset $a' = a + \mathcal{N}(0, \sigma * a)$, where $\mathcal{N}(0, \sigma * a)$ is a normal distribution noise with zero mean and variance $\sigma * a$. Similarly, the actual return point $\mathbf{N}'$ is computed as $(x', y') = (x, y) + \mathcal{N}(0, \sigma * b)$, where (x, y) are the coordinates of $\mathbf{N}$ and (x', y') of $\mathbf{N}'$. Here, the variances of the noise distributions affecting the heading change point and the return point are controlled by $\sigma * a$ and $\sigma * b$, where σ is the parameter governing the uncertainty level. This uncertainty model implies that a less deviated and immediately implemented maneuver suffers less from the environmental uncertainties, while a maneuver with large deviation and further in time suffers more.

D. Scenario representation

In RL, it is never too much to emphasize the importance of the environment's state representation, as any decision made by the agent is heavily influenced by the agent's perceived state of its environment, and the state representation determines how the agent apprehends the state. In the given problem, to ensure that actions taken by the agent always modify the separation status of the ownship, the scenario representation must encapsulate the ownship's current separation status. Therefore, it is reasonable, and also important, to include the CPA closure vectors $\overrightarrow{d_i}$ in the state vector, because these vectors carry the essential information on the separation statuses of the ownship with other aircraft in the environment. With this in mind, we design the one-dimensional state vector s to represent a scenario as follows.

- The first element is σ indicating the current environmental uncertainty level.
- Separation statuses between the ownship and each other aircraft are encapsulated by every 5 next elements:
  - x- and y- positions of the ownship at CPA (2 elements)
  - CPA closure $\|\overrightarrow{d_i}\|$ (1 element)
  - x- and y- directions of the CPA closure vector (2 elements)
- Directional guidance vector (the last 2 elements). This vector is chosen to be $\overrightarrow{\mathbf{NO}}_{\text{CPA}}$, where $\mathbf{N}$ is the return point and $\mathbf{O}_{\text{CPA}}$ the location at CPA of the ownship against the intruder at the beginning. We shall discuss this in the definition of reward function below.

Note that the total length of the state vector depends on the maximum number of aircraft being considered.

E. Maneuver reward

The reward mechanism is designed to give merit to any maneuver suggested by the agent that successfully separates the aircraft and to punish one that fails to improve the separation status. The environment evaluates the reward based on the resultant state of the scenario upon implementation of the suggested maneuver. As the ultimate aim is to separate the aircraft, more positive rewards are given to maneuvers that improve the separation status, while maneuvers that worsen the situation are punished by negative reward. Furthermore, for a valid maneuver that successfully resolves the conflict, the quality of the maneuver is also evaluated, such as deviation from the original trajectory and maneuverability of the resolution.

Let $R(a, s')$ being the reward function that takes an action a together with its resultant state vector s' as two input

arguments and returns the reward value. Also, we denote $d_{\text{min}} = \arg\min_i \|\overrightarrow{d_i}\|$, $(1 \leq i < n)$, as the minimum value among all the separation distances of the ownship against other aircraft. Then, the reward function is defined as

$$R(a, s') = \begin{cases} e^{d_{\text{min}}-1} - 1, & \text{if } d_{\text{min}} < d_{\text{sep}} \quad (1) \\ (1 - \dfrac{\Delta D}{\Delta D_{\text{max}}}) * 100, & \text{otherwise} \quad (2) \end{cases}$$

where ΔD denotes the deviation of the maneuver from the original ownship's trajectory, and ΔD_{max} the maximum deviation that could occurs. In the definition of the reward function, Eq. 1 punishes invalid maneuvers that cause $d_{\text{min}} < d_{\text{sep}}$ and therefore fail to separate the ownship from other aircraft. On the other hand, Eq. 2 calculates the rewards for valid maneuvers, which successfully separate the ownship and eliminate all potential conflicts, by evaluating the deviations of the maneuvers from the ownship's original trajectory. Less deviated maneuvers receive higher rewards, on a score scale of maximum 100. The maneuver's deviation is defined as

$$\Delta D = w_1 * dist(\mathbf{M}, \mathbf{O}_{\text{CPA}}) + w_2 * dist(\mathbf{N}, \mathbf{O}_{\text{CPA}}), \quad (3)$$

where $dist()$ yields the distance between two points. Here, Eq. 2 and Eq. 3 imply that less deviated maneuvers shorten the distances $\mathbf{MO}_{\text{CPA}}$ and $\mathbf{NO}_{\text{CPA}}$. This reflects the design of the reward mechanism to maintain the positions of the action points ($\mathbf{M}$ and $\mathbf{N}$) within the neighborhood of the initial conflict location, which could help preventing the maneuver from causing secondary conflicts with surrounding aircraft. This also justifies the inclusion of $\overrightarrow{\mathbf{NO}}_{\text{CPA}}$ in the state representation as mentioned in section II-D.

III. AI Agent and Learning Mechanism

In our problem, the ultimate goal is to train the AI agent such that given a conflict scenario, it could resolve the conflict and earn a possibly highest reward after a finite number of actions, as quickly as possible. Instead of using classical optimization approaches, here, we adapt the Deep Deterministic Policy Gradient (DDPG) algorithm [26] for our learning model. In this section, we briefly describe our AI agent, show the characteristics of the proposed DDPG algorithm that make it appropriate for training the agent, and discuss the training process as well as some implementation considerations.

A. Agent's Action for Reinforcement Learning

When resolving a conflict, the agent could suggest a possible maneuver by computing the set of three parameters (t_1, α, t_2) that fully defines the maneuver, as mentioned in section II-B. We could see from Figure 3 that any value of (t_1, α, t_2) is equivalent to a choice of (t, x, y), where $t = t_1$ is the heading change time, x and y are the coordinates of the return point $\mathbf{N}$, relatively to the center of the interested area. Moreover, the possible valid choices of (x, y) highly depend on t; therefore, it is rational to treat the agent's action as a two-stage decision-making process. In the first stage, the agent determines the heading change time t that results in the heading change point $\mathbf{M}$. In the second stage, it decides the coordinates (x, y) of the return point $\mathbf{N}$, being aware of the updated aircraft's locations at time t. This treatment of the agent's action is beneficial in two ways. First, as t and (x, y) are different in nature, the two-stage process allows us to handle them independently. Second, such approach avoids the computing of the original parameters (t_1, α, t_2) using the same model, which could be problematic because they might be very different in scale.

Thus, reward for a conflict scenario can be defined as:

$$V(s) = R(t|s) + R((x, y)|t, s) \quad (4)$$

In which $R(t = t_i|s)$ is the immediate reward for selecting $t = t_i$ as time duration and $R((x,y)|t = t_i, s)$ is the reward for selecting the return position *(x,y)* given previous decision $t = t_i$ and conflict scenario s. Therefore, the optimal reward for a given scenario is:

$$\begin{aligned} V^*(s) &= \max_{t,(x,y)} (R(t|s) + R((x, y)|t, s)) \\ &= \max_t [R(t|s) + \max_{(x,y)} R((x, y)|t, s)] \end{aligned} \quad (5)$$

We apply the principle of dynamic programming to obtain the last equation in Equation 5. The Equation 5 shows how we convert this problem from finding a 3-dimensional maneuver into finding the time duration t. For each value of t, we always compute the maximum reward over a set of possible return positions (x, y). Then finding optimal value for given scenario is equivalent to search value of t^* to maximize the reward. Additionally, the ultimate goal is to recommend the best maneuver $m^* = (t^*, (x, y)^*)$ for a given conflict scenario. Besides value of t^*, we also need to get the return point $(x, y)^*$ which provides the optimal value for $R^*((x, y)|t, s)$. An actor-critic algorithm (1) is a good candidate for modelling the second part, $R((x, y)|t, s)$, since it can provide both optimal values (Q^*_value) and optimal maneuver ($(x, y)^*$) at the same time. The propose approach is presented in Figure 5 in detail. For a given scenario s', and a time duration t (<Time to CPA), a time-shifted scenario s is computed. This conflict scenario s is the input for an actor-critic model which provides optimal maneuver $m'^* = v(x, y)^*$ and optimal value $Q^*(s, m') = Q^*(s, (x, y)|s', t)$. To simplify the problem, t is a set of discrete values $(t_0, t_1, ..., t_N)$ which can be looped over to find the optimal value. Moreover, in air traffic control domain, the deterministic characteristic (2) of decision is important for any model i.e. given a conflict scenario, we always receive the same recommended maneuver (without retraining the model). Finally, the possible space for return point is large and continuous in nature (3) which is a challenge for several RL models. (1),(2) and (3) are the reason on how we select Deep Deterministic Policy Gradient (DDPG) algorithm [26] as the actor-critic model.

The next challenge for applying DDPG is to define the *action* for AI agent. The *action* for an AI agent can be defined in several ways which shape the learning mechanism and affect agent strategy. For instance, the agent's action can be the same as the maneuver $a \equiv m' \equiv (x, y)$ or similar to the searching step in which agent searches around

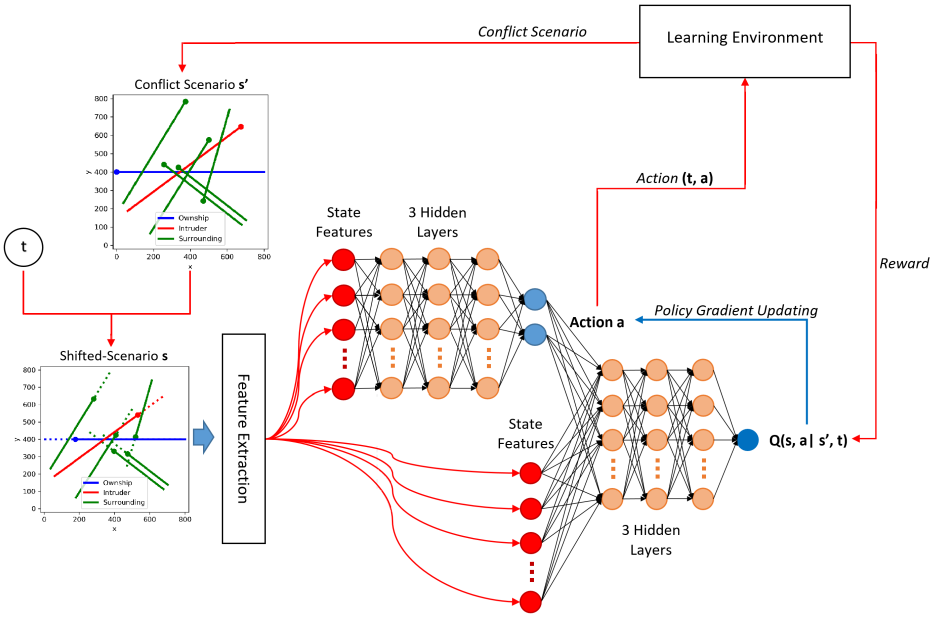

Figure 5. Model for learning conflict resolution.

in multi-steps for the good return point position, etc. In this study, we apply the second method. The agent's action is a moving step (dx, dy), and the agent performs a sequence of actions $[(dx_0, dy_0), (dx_1, dy_1), ..., (dx_k, dy_k)]$ $(dx, dy \leq Radius(rd))$ to optimize the location (x, y) of the return point **N**. Let $(x, y)^* \equiv (x^*, y^*)$ denote the optimal location of **N**, where $x^* = x_0 + \sum_{i=0}^{k}(dx_i)$ and $y^* = y_0 + \sum_{i=0}^{k}(dy_i)$; in which the values of the $Radius(rd)$ is provided in part IV and the value of each action, the number of action are controlled by the learning algorithm and the learning mechanism for AI agent is described in detail in III-C.

B. Deep Deterministic Policy Gradient (DDPG)

DDPG is a variant of actor-critic model based on Deterministic Policy Gradient(DPG) algorithm [25]. One of its main contributions is introduction of a neural network as actor model to deal with continuous action space. The DDPG algorithm has two models:

1) Actor Model: This is a neural network for learning the mapping from state to action, $\mu(s)$. Given a state feature vector, described in II-D, actor model will predict an optimal action a^* under current policy. In this case, given the state vector, actor model will predict the moving action $a_i^* = (dx_i, dy_i)^* = \mu(s_i)$ to update the current maneuver under current policy.
2) Critic Model: This is neural network to evaluate the quality of action given conflict scenario. It receives scenario s_i and action a_i as inputs and estimates the expected value/reward $Q(s_i, a_i)$.

As mentioned in III-A, the main learning algorithm is DDPG but we also introduce a searching step like in Deep Q-Learning to identify the heading change time t. Our proposed 2-stages action DDPG is described in Algorithm 1. Our implementation also consider following enhancements in DDPG such as:

- Replay Memory: to store pass experiences for batch training which can solve the problem about dependence of samples, predicted maneuvers in our cases. The training process begins only when the Replay Memory has been filled with a minimum number of samples. The memory's capacity (i.e. the maximum number of samples in the Replay Memory) is fixed, and this helps to eliminate out-of-date samples in the training process.
- Batch Normalization: to deal with multiple units and ranges in input scenario.
- *Soft target update* is used to increase the stability of learning. Line 16 in Algorithm 1 shows how to apply soft target. The learning rate or update rate is control by parameter τ
- Action in RL is considered to balance between exploration and exploitation. DDPG allows to separate exploration from the algorithm by introducing a new noise policy $\mu_{\mathcal{N}} = \mathcal{N} + \mu$ as exploring policy where exploring noise $\mathcal{N}$ is a random process. Ornstein-Uhlenbeck Noise (OU noise) is implemented as our exploring noise as in [26]. Figure 6 shows examples for exploring search path with 10 steps with OU noise. The set of parameters for exploring noise (OU noise) is $(\mu_e, \theta_e, \sigma_e)$

C. Learning Mechanism

The interaction between AI Agent and Learning Environment is the core mechanism for training and testing for RL as in Figure 5. Since the conflict resolution is a continuous control problem, thus the episode should be different from the classical time-based episode. The episode is designed

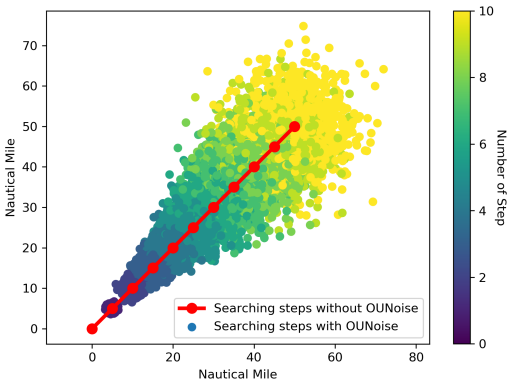

Figure 6. Visualization of exploratory search path with Ornstein-Uhlenbeck noise given a constant search path (10 steps with constant step-size is 5 NM).

Algorithm 1: DDPG Algorithm for 2-stages action

Randomly initialize weight θ^Q for Critic Net $Q(s, a|\theta^Q)$
Randomly initialize weight θ^μ for Actor Net $\mu(s|\theta^\mu)$
Initialize target networks Q' and μ' by $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$.
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration.
  Receive scenario s' from Environment
  Computing new scenario s_1 by shifting all flights in s' a duration $heading\_change\_Time$ $t_0 = random(0, Max\_T)$
  for t = 1, Max_steps **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to current policy.
    Execute action a_t, observe reward r_t and new state s_{i+1}
    Store transition s_t, a_t, r_t, s_{t+1} in R
    Sample a random K experiences s_i, a_i, r_i, s_{i+1} from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss:

$$L = \frac{1}{K}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$$

    Update actor policy using sampled policy gradient:

$$\nabla_{\theta^\mu}(J) \approx \frac{1}{K}\sum_i \nabla_\mu Q(s_i, \mu(s_i)|\theta^Q)\nabla_{\theta^\mu}\mu(s_i|\theta^\mu)$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

as a searching process to locate an "acceptable resolution" (examples can be observed in Figure 7). At each step, the agent will predict the best action (dx, dy) to modify the current resolution and send it to environment. Learning environment will update the current maneuver $(x', y') = (x + dx, y + dy)$, evaluate it and send feedback back to agent. The process is repeated until receiving acceptable resolution or the number of searching steps is reached. Acceptable resolution can be flexibly defined by controlling the threshold for minimum reward of acceptable resolution. Without satisfying those stopping conditions, the steps are considered as intermediate steps and their rewards are constant number x (= -0.1) as a small penalty (Figure 8).

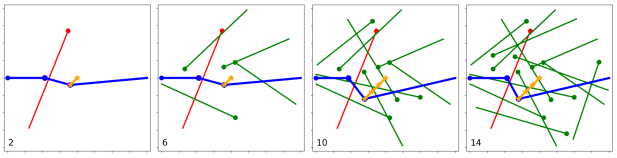

Figure 7. Examples of a searching episode to suggest resolution

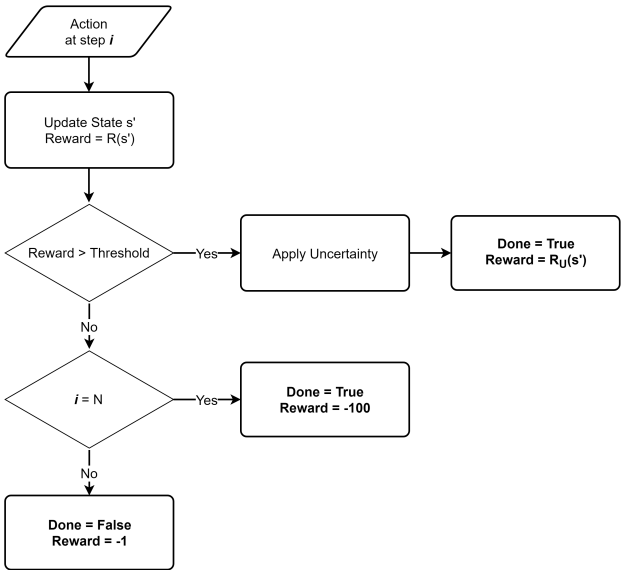

Figure 8. Flowchart to compute reward for each resolution

Algorithm 1 and Figure 5 can be used to describe training and testing process for AI Agent in detail. The main purposes of training phase include generating learning samples, training actor and critic target network using DDPG algorithm. While in testing phase, given an unseen scenario, only actor target network is used to predict "optimal" maneuver. The step by step algorithm for training is as follows.

1) Learning Environment generates random scenario s'.
2) The conflict scenario s' is shifted with random heading change, time t_0 to obtain shifted conflict scenario s_0.
3) Feature extraction algorithm is applied on Shifted conflict scenario s_t (t is initialized by 0 for each new scenario) to obtain state vector which is the input for DDPG algorithm.
4) Given the state vector, current exploration actor policy (current actor model $\mu(s_t|\theta^\mu)$ plus exploration noise $\mathcal{N}_t$) will provide a candidate action $a_t \equiv (dx, dy)$.
5) The action is sent to learning environment to compute the real reward r. If the conditions for stopping episode are reached, the uncertainty model will create noise-action by adding random noise to proposed action and then compute the reward r_t for it. Else, environment just return immediate constant reward $r_t = 0.1$ and updated scenario s_{t+1} after applying action to modify the current maneuver.
6) The sample tuples (s_t, a_t, r_t, s_{t+1}) is stored in replay buffer for later use in training model.

7) When the replay buffer has stored enough samples ($\geq$ minimum start size), a batch of samples is sampled randomly from replay buffer for training.
8) The critic model is updated by minimizing the defined loss function which is similar to training supervise learning model.
9) The policy gradient is computed from the gradient of critic model and applied to update actor model at each step.
10) Finally, the target networks are updated in soft updating manner.
11) If end of episode is reached, the searching step will be stopped and go back to step 1. Else, increase $t = t + 1$ and go back to step 3.

The testing phase or predicting phase is relative simple since we only need to obtained the final recommended maneuver for given conflict scenario. However, in practical use, the experiences generated in this phase can also be stored in replay buffer for tuning the model via batch training. This setting can help the model tuning to be faster and keep the model up-to-date with new incoming data. The step-by-step for maneuver prediction is described as follows.

1) Given unseen scenario s' (i.e. from learning environment):
2) The heading change time t is looped over the range $[T_{min}, T_{max}]$ with step-value Δt seconds. The conflict scenario s' is shifted with each given heading change time t to obtain shifted conflict scenario s.
3) Feature extraction algorithm is applied on shifted conflict scenario s_i to obtain state vector which is the input for actor model.
4) Given state vector, actor target model suggests action $a_i^* \equiv (dx_i, dy_i)^*$.
5) The action is sent to the environment. If end of episode is reached, go to next step, else compute next state s_{i+1}, $i = i + 1$ and return to step 3.
6) Given state vector and "optimal" action, critic target model will provide the Q-value $Q^*(s, a^*|t)$.
7) Compute $MAX\_Q = \max_t Q^*(s, a|t)$ and store corresponding maneuver $m = (t, (x, y)^*)$.
8) Finally, after checking with all values of heading change time t, the "optimal" maneuver for given conflict scenario s' is obtained $(t^*, (x, y)^*)$

IV. Experiment Setup

In our experiments, conflict scenarios are randomly generated in an interested area of radius $r = 50$ nm. For the initial conflict, $d_{1(\text{CPA})} < d_{\text{sep}}$ where $d_{\text{sep}} = 5$ nm, and $240 \leq t_{\text{CPA}} \leq 480$ seconds, given that the common speed of aircraft $v_c = 400$ knots (nm/hr). This configuration implies that the potential loss of separation between two aircraft is foreseen 4-8 minutes. We consider the maximum number of aircraft in the airspace $n_{\text{max}} = 15$; therefore, the state vector has fixed size of 73 (see section II-D for state representation). In the event the number of aircraft is less than n_{max}, the elements representing the absent aircraft are replaced by that of the intruder. During maneuvers implementation, we consider four levels of environmental uncertainty, $\sigma = \{0, 2\%, 5\%, 10\%\}$.

Figure 9 shows examples of 24 groups of initial conflicts generated in our experiments, consisting of 4 groups of t_{CPA} (time to CPA) and 6 groups of conflict angle ϕ (see Figure 2 for the definition of conflict angle). This classification allows us to assess the model's performance in different classes of initial conflicts. Note that surrounding traffic are not shown in Figure 9.

Parameters used for training the agent are shown in Table 1. The values of these hyper parameters are chosen from practise. For example, γ ([0,1]) is the discount factor to weight the importance for future rewards. If $\gamma \leftarrow 0$, the agent will focus only on immediate rewards. Else, if $\gamma \leftarrow 1$, the future rewards have greater weight in the model. In our case, final step of episode is the required maneuvers which is the main source of reward, thus in this study $\gamma = 1$.

TABLE I
Parameters for training the AI agent

Parameter	Meaning	Value
lr_{actor}	Control learning rate of actor model	10^{-4}
lr_{critic}	Control learning rate of critic model	10^{-3}
$batch\_size$	Size of training batch	64
γ	Discount factor for future rewards	1
τ	Control rate of updating target networks for both models	10^{-3}
$\mu_e, \theta_e, \sigma_e$	Parameter set for exploration noise	$0, 0.1, 0.5$
$thres_{maneuver}$	minimum acceptable reward	40
Max_steps	Maximum number of searching steps	10
$Radius(rd)$	Upper bound of agent's action	5NM
Δt	Time step for heading change time	30 seconds

We train the RL Model by interacting with our environment and learn from those experiences. At each 500 iterations, learning environment will generate 4800 random scenarios (200 scenarios from each group) for evaluating current model. Each conflicted scenario then is assigned a randomly number of aircraft (2 to 15) and uncertainty level (4 levels). This setup tries to limit the computational cost for intermediate evaluation.

V. Results and Discussion

We first assess the model's performance by the average score (on the scale of 100) that the agent earns for solving 4800 unseen scenarios of a common test set. Figure 10 shows the test results (or model's convergence) as the model is evolving during training at different levels of environmental uncertainty (subplots (a) for no uncertainty, (b) for 2% , (c) for 5%, and (d) for 10%). Each data point represents the average score performed on the common test set after every 500 training iterations. Each curve reports the result for a different number of aircraft n (blue curve for $n = 2$, orange for 5, green for 10 and red for 15).

A common trend is observed from Figure 10 that for all configurations, the training curves admit three distinguished phases, approximately: the warming up phase in the first 500 iterations, the evolving phase in the next 1500 interactions, and

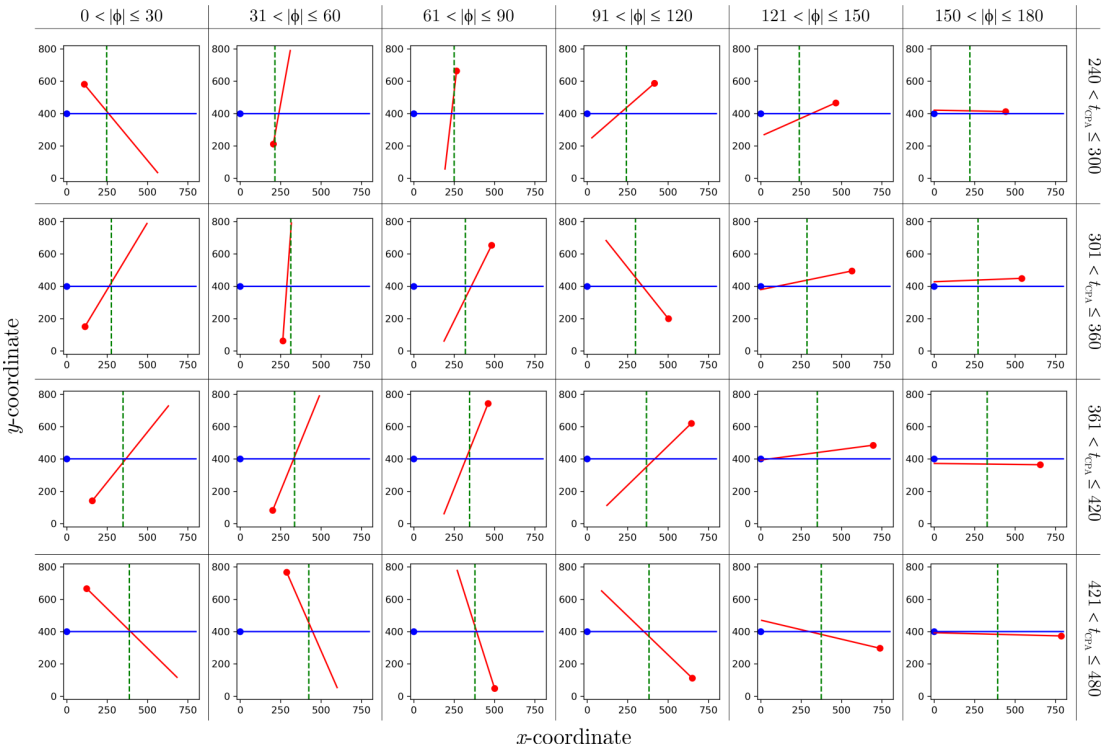

Figure 9. Generated scenarios are classified into 24 groups, based on their time to CPA (t_{CPA}, across the rows) and conflict angle (ϕ, across the columns).

the converging phase after 2000 iterations. The observed trend in the performance curves suggests that our current setting is able to guarantee the model's convergence at different uncertainty conditions and numbers of aircraft.

On the other hand, the different impacts of environmental uncertainty σ and total number of aircraft n on the learning performance are also observed. One could observe from Figure 10 that the environmental uncertainty has stronger impact on the convergent speed than the number of aircraft does. Higher levels of the environmental uncertainty more delay the model's convergence. In particular, at low levels of uncertainty, i.e. $\sigma = \{0, 2\%\}$, convergence occurs after about 1500 iterations, while at higher levels, i.e. $\sigma = \{5\%, 10\%\}$, the model only starts to converge when the training has reached about 2000 iterations.

The environmental uncertainty not only reduces the convergent speed, but it also affects the stability of the convergent score. Figure 11a plots the variances of the achieved scores at different levels of uncertainty. We can see that variances increases with the uncertainty level, suggesting that the model's performance is less stable at higher uncertainty. Figure 11a also shows how the convergent stability is reduced by increasing the number of aircraft. At this point, it is meaningful to plot the variances being normalized with that at no uncertainty, as shown in Figure 11b; this allows us to observe the increasing rate of variances when the environment becomes more uncertain. For instance, at the number of aircraft $n = 2$, introducing 2% uncertainty causes the score variance increased by about 2.8 times, 5% uncertainty increases the variance by 4 times, and 10% uncertainty results in 5 times increase. The fact that the growth rate of the variances (with increasing uncertainty) is lower at higher number of aircraft, as seen from 11, suggests that the model could control the total score variance caused by the increasing in both uncertainty and the number of aircraft.

Figure 12 provides a closer look to the performance of the model after convergence, indicated by the average reward and the successful rate. We define the successful rate as the percentage of the conflicts being successfully resolved by the agent with a reward higher than an acceptable threshold (40 out of 100 in our setting). From Figure 12, we observe linear relations between the model's performance indicators (i.e. score and successful rate) and the number of aircraft involved. Increasing the number of aircraft cause the performance to drop, and the environmental uncertainty even worsens this drop. For instance, by increasing the number of aircraft from 2 to 15, the successful rate drops by 1% (from $\approx 99\%$ to $\approx 98\%$) at no uncertainty, and by 10% (from $\approx$ 91% to $\approx$ 81%) at 10% uncertainty. Similar trend is also observed in the change of reward with the increasing number of aircraft. The results reported in Figure 12 indicate the significant impact of the number of aircraft on the performance of the agent; nevertheless, the successful rate of about 81% achieved by our agent at high uncertainty ($\sigma = 10\%$) and dense surrounding traffic ($n = 15$) proves the high performance of the learning model.

Figure 13 presents the maneuvers suggested by the agent for resolving 3 conflicts (each in a row) at different numbers

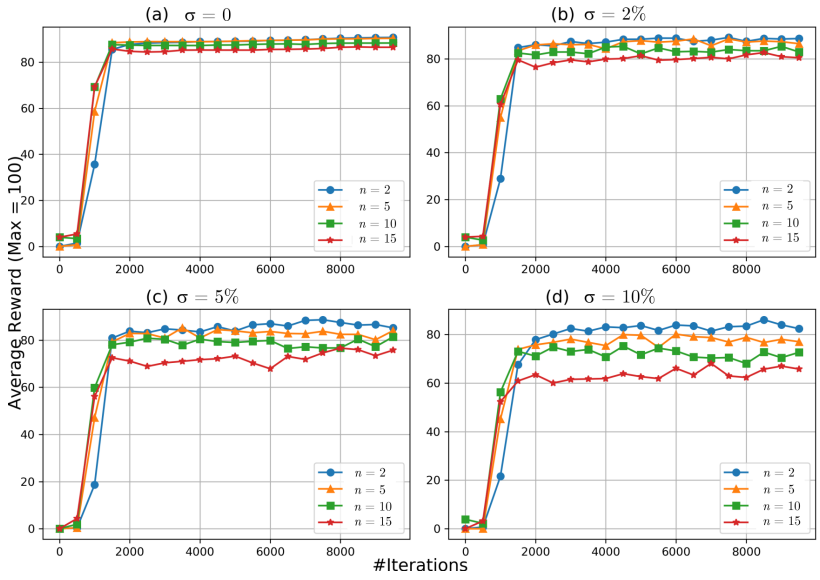

Figure 10. Convergence of learning model at different configurations of uncertainty level (σ) and number of aircraft (n)

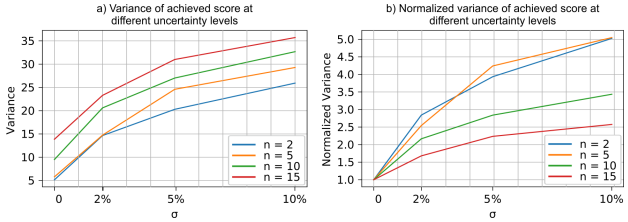

Figure 11. Effects of the uncertainty on the performance stability after convergence.a) Variances of achieved scores. b) Variances being normalized with variances at no uncertainty.

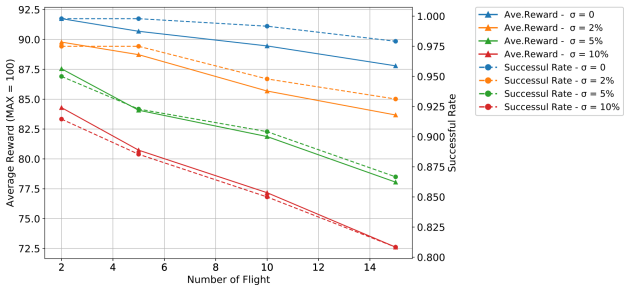

Figure 12. Average reward and successful rate achieved by the agent after convergence.

of surrounding aircraft. The blue gradients in the backgrounds represent the feasible regions of the maneuvers' return points (at pre-determined heading change points). Maneuvers with return points located in the darker regions receive higher rewards. As the number of aircraft increases, the feasible region fragments into more disconnected smaller regions. The fragmentation of the feasible region causes a drop in model's performance (Figure 12), especially under high uncertainty, because strong environmental disturbance more possibly shifts the agent's suggested return point from a feasible region to a

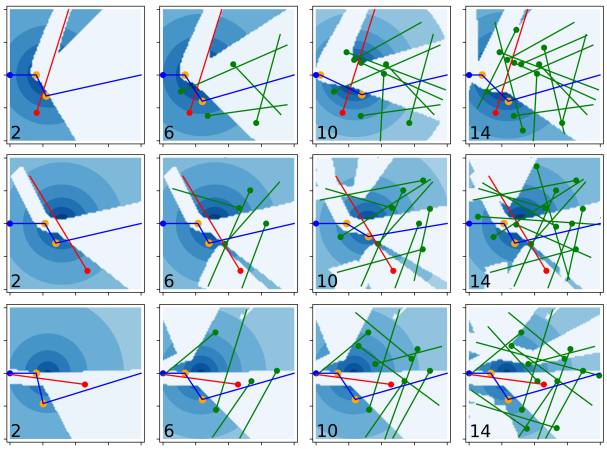

Figure 13. Examples of predicted resolutions in scenarios with different number of flights

impractical location.

Another interesting observation from Figure 13 is the constructed strategy of AI Agent to resolve conflict scenario. The searching steps is always going down, thus, it is equivalent the flight will turn right at the heading change point. This strategy is simple and although it provides high quality solution, from Figure 13, we notice that it isn't global optimal but just a local optimal. This phenomena can be a results of several reasons but one of the main reason is the equalization of turning left and turning right as resolutions for conflict scenario as well as the design of reward itself.

Finally, in Figure 14, we perform the assessment on how our model can learn and approximate real reward from environment given a conflict scenario. The red line is "the optimal line" where the approximation (Q_value) is exactly the same as read reward. As showed in the figure, the approximated

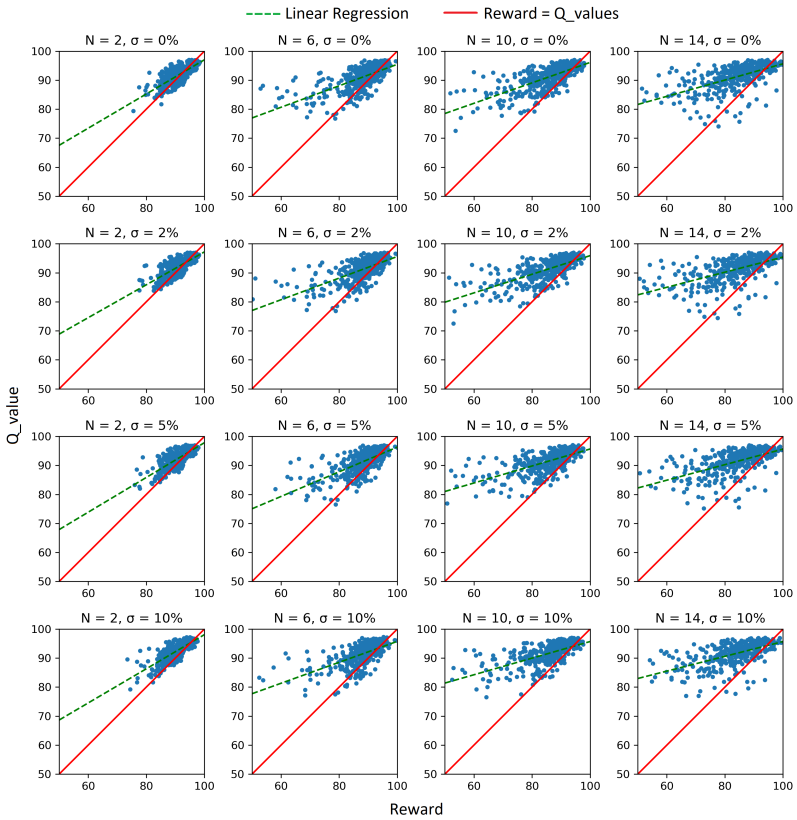

Figure 14. The approximation of Q_value to real Reward when Number of Flights (NoF) from 2 to 14 and Uncertainty (U) from 0% to 5%

values is higher than real rewards in most of the case. We apply simple linear regression to capture the linear relation between Q_values and Rewards. The green dash line shows the tendency of over-estimated reward of the proposed model. It isn't affected much by the uncertainty but the number of flight in scenario. It reflects the difficulty of high traffic scenario in learning approximator because of its diversity. However, the regression lines also report the linear relation between real rewards and Q values. This relation is important for training the actor model because actor model needs the relative scores between different Q(s,a) pairs for choosing best action rather than their exact values. This means that given current level of approximation, the performance of actor won't be affected.

VI. Conclusion

In this work, we have formulated the problem of conflict resolution in the presence of surrounding traffic and uncertainty as a reinforcement learning problem. Important components of the reinforcement learning algorithm for conflict resolution, such as learning environment, scenario state representation, reward function, and learning algorithm, have been discussed in great details. We have also laid out the evaluation of model's performance, which could be considered as a framework for the assessment of reinforcement learning method applied to conflict resolution problem. Our findings show that the combination of Deep Q-learning and Deep Deterministic Policy Gradient algorithms gives the AI agent the great capability to suggest high quality conflict resolution, with a successful rate of over 81% in the presence of dense surrounding traffic and strong environmental disturbance. Here, it should be highlighted that the agent achieved this high successful rate without the need of prior knowledge about a set of rules mapping from conflict scenarios to expected actions. The successful rate of the algorithm could be further improved during the interaction between the AI agent and the controllers. In particular, feedback from the controller, i.e. accepting or rejecting agent's resolution, could be collected to train the agent in order to enhance its experience. Furthermore, the objective function could dynamically evolve to reflect the controllers' preference when the AI agent is exposed to new feedback data provided by the controllers. Our system could be applied in training novice controllers, in which the agent can reproduce and recommend resolutions for trainee controllers to observe and learn. Possible future considerations to improve the system include but not limited to (1) the enhancement of the scenarios state representation to help the agent to better "perceive" its learning environment and (2) the extension of the current work to multi-agent system for cooperative conflict resolutions.

VII. Acknowledgement

This research is partially supported by Air Traffic Management Research Institute (NTU-CAAS) Grant No. M4062429.052

References

[1] IATA, "20 year passenger forecast," https://www.iata.org/publications/store/Pages/20-year-passenger-forecast.aspx, 2018, [Accessed 27-December-2018].

[2] J. K. Kuchar and L. C. Yang, "A review of conflict detection and resolution modeling methods," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 4, pp. 179–189, 2000.

[3] Y. Yang, J. Zhang, K. Cai, and M. Prandini, "Multi-aircraft conflict detection and resolution based on probabilistic reach sets," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 1, pp. 309–316, 2017.

[4] S. Hao, S. Cheng, and Y. Zhang, "A multi-aircraft conflict detection and resolution method for 4-dimensional trajectory-based operation," *Chinese Journal of Aeronautics*, vol. 31, no. 7, pp. 1579–1593, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1000936118301705

[5] N. Yokoyama, *Decentralized Conflict Detection and Resolution Using Intent-Based Probabilistic Trajectory Prediction*, ser. AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, 2018. [Online]. Available: https://doi.org/10.2514/6.2018-1857

[6] V. P. Jilkov, J. H. Ledet, and X. R. Li, "Multiple model method for aircraft conflict detection and resolution in intent and weather uncertainty," *IEEE Transactions on Aerospace and Electronic Systems*, pp. 1–1, 2018.

[7] M. Radanovic, M. A. Piera Eroles, T. Koca, and J. J. Ramos Gonzalez, "Surrounding traffic complexity analysis for efficient and stable conflict resolution," *Transportation Research Part C: Emerging Technologies*, vol. 95, pp. 105–124, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X18302353

[8] C. Allignol, N. Barnier, N. Durand, A. Gondran, and R. Wang, "Large scale 3d en-route conflict resolution," in *ATM Seminar, 12th USA/Europe Air Traffic Management R&D Seminar*, Conference Proceedings.

[9] Z. Liu, K. Cai, X. Zhu, and Y. Tang, "Large scale aircraft conflict resolution based on location network," in *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, Conference Proceedings, pp. 1–8.

[10] S. Ravizza, J. Chen, J. A. Atkin, P. Stewart, and E. K. Burke, "Aircraft taxi time prediction: comparisons and insights," *Applied Soft Computing*, vol. 14, pp. 397–406, 2014.

[11] H. Lee, W. Malik, and Y. C. Jung, "Taxi-out time prediction for departures at charlotte airport using machine learning techniques," in *16th AIAA Aviation Technology, Integration, and Operations Conference*, 2016, p. 3910.

[12] M. Ahmed, S. Alam, and M. Barlow, "A cooperative co-evolutionary optimisation model for best-fit aircraft sequence and feasible runway configuration in a multi-runway airport," *Aerospace*, vol. 5, no. 3, p. 85, 2018.

[13] S. Ayhan and H. Samet, "Aircraft trajectory prediction made easy with predictive analytics," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 21–30.

[14] R. Alligier, D. Gianazza, and N. Durand, "Machine learning and mass estimation methods for ground-based aircraft climb prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, pp. 3138–3149, 2015.

[15] M. Conde Rocha Murca, R. DeLaura, R. J. Hansman, R. Jordan, T. Reynolds, and H. Balakrishnan, "Trajectory clustering and classification for characterization of air traffic flows," in *16th AIAA Aviation Technology, Integration, and Operations Conference*, 2016, p. 3760.

[16] N. Takeichi, R. Kaida, A. Shimomura, and T. Yamauchi, "Prediction of delay due to air traffic control by machine learning," in *AIAA Modeling and Simulation Technologies Conference*, 2017, p. 1323.

[17] S. Choi, Y. J. Kim, S. Briceno, and D. Mavris, "Prediction of weather-induced airline delays based on machine learning algorithms," in *Digital Avionics Systems Conference (DASC), 2016 IEEE/AIAA 35th*. IEEE, 2016, pp. 1–6.

[18] Y. J. Kim, S. Choi, S. Briceno, and D. Mavris, "A deep learning approach to flight delay prediction," in *Digital Avionics Systems Conference (DASC), 2016 IEEE/AIAA 35th*. IEEE, 2016, pp. 1–6.

[19] J. Schaeffer, "A gamut of games," *AI Magazine*, vol. 22, no. 3, p. 29, 2001.

[20] M. Campbell, A. J. Hoane Jr, and F.-h. Hsu, "Deep blue," *Artificial intelligence*, vol. 134, no. 1-2, pp. 57–83, 2002.

[21] N. Yakovenko, L. Cao, C. Raffel, and J. Fan, "Poker-cnn: A pattern learning strategy for making draws and bets in poker games using convolutional networks." in *AAAI*, 2016, pp. 360–368.

[22] M. Moravčík, M. Schmid, N. Burch, V. Lisỳ, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, and M. Bowling, "Deepstack: Expert-level artificial intelligence in heads-up no-limit poker," *Science*, vol. 356, no. 6337, pp. 508–513, 2017.

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[24] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[25] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *ICML*, 2014.

[26] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

AUTHOR BIOGRAPHY

Duc-Thinh Pham obtained his M.Sc. degree in Computer Science from Telecom ParisTech, France, in 2013. He is a PhD candidate of Paris Science and Letter (PSL) in Complex system. His research focus on applying machine learning algorithms for predicting capability in air traffic control.

Ngoc Phu Tran received his PhD in Mechanical Engineering from Nanyang Technological University (NTU), Singapore in 2017. He is currently a research fellow in AI & Data Analytics research group at Air Traffic Management Research Institute (ATMRI), Nanyang Technological University, Singapore. His research focuses on the design of artificial intelligence algorithms for air traffic management.

Sameer Alam is the Programme Director of AI & Data Analytics at Air Traffic Management Research Institute (ATMRI) and an Associate Professor at the School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore. He obtained his PhD in Computer Sc. from University of New South Wales (UNSW), Australia in 2008.

Vu Duong is a Professor and Director of Air Traffic Management Research Institute (ATMRI), School of Mechanical & Aerospace Engineering, Nanyang Technological University, Singapore. Vu had also been Head of Innovative Research then Senior Scientific Advisor at EUROCONTROL (1995-2012), and a member of SESAR JU Scientific Committee (2010-2012). He obtained his PhD in Artificial Intelligence from Ecole Nationale des Ponts et Chaussees, France in 1990.

Daniel Delahaye is a full Professor and Director of the OPTIM research laboratory of ENAC, Toulouse, France. He obtained his Ph.D in automatic control from the Aeronautic and Space National school, Toulouse in 1995 and did a post-doc at the Department of Aeronautics and Astronautics at MIT in 1996.