

Deep Reinforcement Learning based Path Stretch Vector Resolution in Dense Traffic with Uncertainties

Daniel Delahaye, Duc-Thinh Pham, Phu N Tran, Sameer Alam, Vu Duong

▶ **To cite this version:**

Daniel Delahaye, Duc-Thinh Pham, Phu N Tran, Sameer Alam, Vu Duong. Deep Reinforcement Learning based Path Stretch Vector Resolution in Dense Traffic with Uncertainties. Transportation research. Part C, Emerging technologies, 2021. hal-03425669

HAL Id: hal-03425669
https://enac.hal.science/hal-03425669

Submitted on 11 Nov 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Deep Reinforcement Learning based Path Stretch Vector Resolution in Dense Traffic with Uncertainties

Duc-Thinh Pham, Phu N. 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. This gives rise to the need for conflict resolution advisory tools that can perform well in high-density traffic scenarios given a noisy environment. Unlike model-based approaches, learning-based approaches can take advantage of historical traffic data and flexibly encapsulate environmental uncertainty. In this study, we propose a reinforcement learning approach that is capable of resolving conflicts, in the presence of traffic and inherent uncertainties in conflict resolution maneuvers, without the need for 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 a large and complex action space. The research also includes the development of a learning environment, scenario state representation, reward function, and a reinforcement learning algorithm inspired from Q-learning and Deep Deterministic Policy Gradient algorithms. The proposed algorithm, with two stages decision-making process, is used to train an agent that can serves as an advisory tool for air traffic controllers in resolving air traffic conflicts where it can learn from historical data by evolving overtime. Our findings show that the proposed model gives the agent the

capability to suggest high quality conflict resolutions under different environmental conditions. It outperforms two baseline algorithms. The trained model has high performance under low uncertainty level (success rate $\geq$ 95%) and medium uncertainty level (success rate $\geq$ 87%) with high traffic density. The detailed analysis of different impact factors such as environment's uncertainty and traffic density on learning performance are investigated and discussed. The environment's uncertainty is the most important factor which affects the performance. Moreover, the combination of high-density traffic and high uncertainty will be the challenge for any learning models.

Keywords: reinforcement learning, air traffic control, deep deterministic policy gradient, conflict resolution, actor-critic, learning environment

1. Introduction

Air traffic control (ATC) plays a crucial role in the air traffic management (ATM) system as it is responsible for maintaining flight 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 [1]. 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 [2], ATCOs will have to deal with increased traffic in their respective sectors. In such a situation, conflict resolution tools are needed to support ATCOs in high-density traffic scenarios, with inherent uncertainties in the environment.

Many mathematical models for conflict resolution have been proposed in the literature. Heuristic methods are studied as one of the potential approaches

for resolving conflict with multiple aircraft as in [3, 4, 5]. A comprehensive review is reported in Kuchar et al. [6]. Some recent works look into enhancing the capability of such automated conflict solvers. For instance, Yang et al. [7] 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 recent research, Hao et al. [8] employed aircraft' reachable space, where conflict resolution scheme accounts for the intent of the aircraft via aircraft' 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 [9] 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 Jilkov et al. [10], 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 [11]. 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 Liu et al. [12] and Allignol et al. [13]. While the work in [12] uses aircraft location network and limits its resolution's maneuver to velocity adjustment only, the model in [13] provides for 3D conflict resolution with limited uncertainty. In [14] authors propose using quantum heuristics to solve a simplified version of the Air Traffic Management conflict-resolution problem for wind-optimal trajectories involving minimum trajectory modifications. 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 (i.e. there is no learning). In this work, we attempt to overcome these drawbacks by considering a reinforcement learning approach for conflict resolution where the learning model does not require prior knowledge of how to efficiently resolve a conflict, and can self-evolve when being exposed to unseen scenarios.

However, for decision-making problems like conflict resolution, their large and continuous state and action spaces are challenging for machine learning methods. 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 [15], Mnih et al. 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. Also, AlphaGo [16], that defeated a human world champion in Go used neural networks that were trained using supervised learning and RL, in combination with a traditional heuristic search algorithm. Two major challenges must be considered when applying DRL in real-world problems like conflict resolution: sample complexity and hyper-parameters sensitivity. The first challenge is common for most of popular on-policy DRL algorithms such as Trust Region Policy Optimization (TRPO) [17], Proximal Policy Optimization (PPO) [18] and Asynchronous Advantage Actor Critic (A3C) [19], because they require new samples at each gradient step. Off-policy algorithms, such as Q-learning based algorithms [15], Deep Deterministic Policy Gradient (DDPG) [20] have shown their advantages in sample efficiency. However, DDPG still face the problem with stability and hyper-parameter sensitivity [21, 22]. To overcome the second

challenge, maximum entropy formulation is a promising approach to provide a substantial improvement in exploration and robustness. Prior work has proposed model-free DRL algorithms that perform on-policy learning with entropy maximization [23], as well as off-policy methods based on soft Q-learning and its variants [24, 25, 26]. However, they generally do not exceed the performance of state-of-the-art off-policy algorithms, such as DDPG. Later, Soft Actor Critic (SAC) [27] is reported to be able to overcome both challenges and provides better performance than DDPG in complex, high-dimensional tasks. In this work, DDPG and SAC will be selected as learning algorithms for our conflict resolution.

Machine learning techniques, specifically reinforcement learning algorithms, have also been employed to build automated system in air traffic control. Focusing on en-route conflict resolution, [28, 29, 30, 31] develop deep distributed multi-agent reinforcement learning frameworks to resolve conflicts by aircraft speed adjustment. Those frameworks are able to identify and resolve conflicts between aircraft in a high-density, stochastic, and dynamic en-route sector with multiple intersections. Besides, several studies apply RL for conflict avoidance and congestion management. [32] presents a RL algorithm to guide an arbitrary number of aircraft every 5 seconds across a three-dimensional, unstructured airspace while avoiding conflicts and collisions. Multi-agent DRL for collision avoidance in dense airspace is also proposed in [33]. In [34], authors propose a deep ensemble multi-agent RL for controlling speed to reduce the traffic congestion, avoid conflict resolution and improve schedule conformation. As observed from the literature, most of the studies on applying deep reinforcement learning for conflict resolution formulate this problem as a continuous control problem in which all aircraft are controlled or navigated at each time step for avoiding all potential conflicts throughout a given airspace [29, 30, 31, 32, 34, 35]. Besides the work of Mollinga et al. [32] which considers three kinds of maneuvers (e.g., speed, altitude, heading), the other ones only focused on speed control for conflict avoidance. More importantly, among those studies, only Guo et al. [31] proposes the approach for considering uncertainty from the environment.

However their uncertainty is still simply modelled as an uniform random noise, applied on each element of the state vector. Furthermore, since all of the above studies are designed for fully autonomous conflict resolutions, those approaches are unsuitable for a decision support tool where any resolution should be provided to ATCO with sufficient look-ahead time (e.g., at least 4 to 8 minutes before the potential conflict happens) for his/her judgement and decision. To achieve this request, the predictability of the traffic scenario under uncertainties within a given airspace is an important factor. Different from mentioned studies, Tran et al. [36] proposed a DRL approach which is trained using controllers' decisions to provide human's preferred path stretch vector resolutions. The work focuses on the ability to suggest ATCO-like decisions rather than solving the potential conflicts, moreover, there is no environment uncertainty considered in their study.

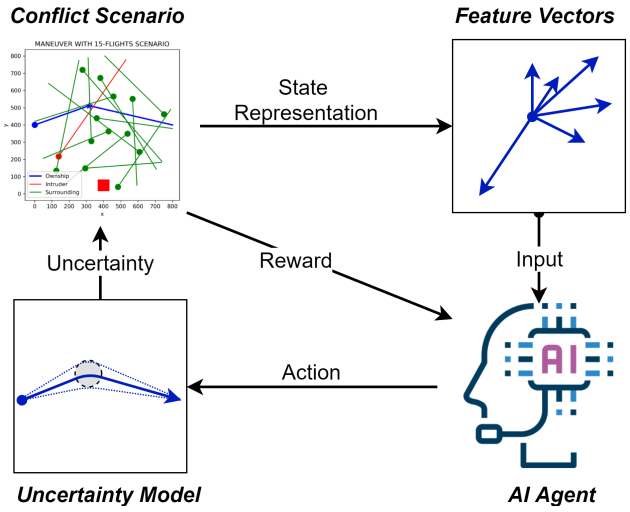


Figure 1: The concept diagram for the interaction between the learning environment and the agent. Conflict scenarios involving multiple aircraft are generated and their features are extracted. The feature vectors are presented to the agent by the learning environment. Then, the agent proposes an action and receives a reward for that maneuver as performance feedback under uncertainty.

Thus, in this study, we propose a novel approach using Reinforcement Learn-

ing for conflict resolution in the presence of surrounding traffic and uncertainty. The algorithm is designed to provide a full path stretch vector resolution (i.e., lateral maneuver) with at least 4 minutes look-ahead time. More importantly, different levels of the uncertainty in implementation of the maneuver are also modelled and considered. Finally, this approach is designed and evaluated while considering some operational constraints as a decision support tool or an advisory tool for ATCO.

The concept diagram of our approach is illustrated in Figure 1. Conflict scenarios involving multiple aircraft are generated and presented to the 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. Noting that the mentioned maneuvers are the the final maneuvers after searching process. 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. Like ATCOs, the agent can learn by resolving conflicts and has the ability to self-evolve via trial-and-error.

In operational environment, the feedback for each maneuver or decision can be accessed only after it is executed. Thus, the classical setting where the uncertainty is implemented in learning environment and return multiple feedback to estimate the expected reward under uncertainty is impractical. To overcome this limitation, our agent will interact with learning environment during searching process but only the final selected resolution is evaluated under uncertainty. This modification increases the challenge for training model under uncertainty, however, it also makes this work more suitable for online learning in operational environment.

The major contribution of this work is to formulate the conflict resolution

problem as a decision-making problem which is then addressed by a RL algorithm. Firstly, an learning algorithm (named DDPG-2S, after the two stage action it uses) is introduced by combining Q-learning and DDPG to handle maneuver actions. Secondly, uncertainty in applying maneuver is introduced in learning environment. Finally, all components in proposed framework are discussed in details for developing reinforcement learning approaches for conflict resolution. In addition, the impact of different factors such as traffic density and uncertainty level on the learning performance are also investigated and discussed in detail. In this study, we give special considerations to the following sub-tasks.

1. Rather than time-based action like continuous control problem, we design our problem such that the agent performs time-independent actions when searching for possible maneuvers.
2. We develop a learning environment for conflict detection and resolution with the following characteristics.
   - The reward function is designed to consider not only the conflict status of the scenarios but also the quality (e.g., deviation, maneuverability, etc.) of the maneuvers.
   - Maneuver's uncertainty is encapsulated in the learning model with different levels.
   - A novel scenarios representation (state vector) is proposed which contains information of the 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).

2. Learning Environment

In 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 conflicts, 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.

The learning environment can perform trajectory prediction and conflict detection to evaluate any maneuver of ownship in term of conflict status and deviation. After finding a good solution, the agent can recommend or apply it to resolve the real flight conflict and get the feedback (or reward) from operational environment. The result or sample will be stored in the agent memory and used to train the model later. The operational environment with uncertainty is mimicked by an function to evaluate maneuver applied uncertainty. Noting that even though applying Monte Carlo can be useful to evaluate quality of maneuver, in real scenarios, we might be able to obtain only one sample for each situation.

2.1. 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. In this study, the density traffic is reflected by the number of flights inside the sector in each conflict scenario. In which, flights' entry points, time or directions are randomly distributed. This setup makes the scenario more complicated than in real operation where the flights are organized into flows. Moreover, in operational

environment, it is unusual to have multiple conflict in the sector for a given short time period. In most of the practical cases, controller needs to resolve one conflict scenario while avoiding creating secondary conflict. Therefore, in this work, 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. With our designed high-density conflict scenario, the challenge is resolving the primary conflict while avoiding all secondary conflicts. This problem is also challenging because of dense traffic.

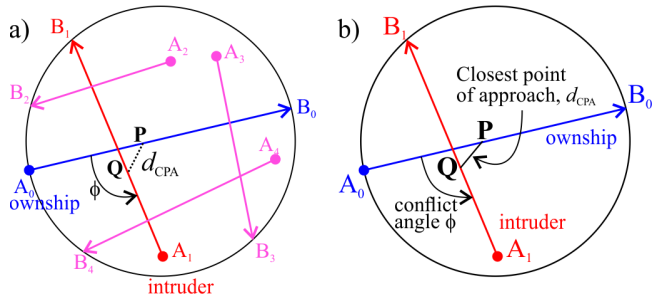


Figure 2: a) Example of a conflict scenario involving a two-aircraft conflict in the presence of four surrounding aircraft. $\mathbf{A}_0\mathbf{B}_0$ is the ownship and $\mathbf{A}_1\mathbf{B}_1$ is the intruder. b) The conflict pair where $\mathbf{PQ}$ is the closest distance between two aircraft.

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 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}, and at the same flight level. 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 $\overrightarrow{u} = v_{\text{c}}(\overrightarrow{\mathbf{A}_0\mathbf{B}_0}/\|\overrightarrow{\mathbf{A}_0\mathbf{B}_0}\|)$ and $\overrightarrow{v} = v_{\text{c}}(\overrightarrow{\mathbf{A}_1\mathbf{B}_1}/\|\overrightarrow{\mathbf{A}_1\mathbf{B}_1}\|)$, respectively. At a time $t > 0$, the locations of the 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 $d_{1(\text{CPA})} = d_1(t_{\text{CPA}})$. Here, it should be noted that when using the aforementioned CPA formula, a positive t_{CPA} is only found if the two aircraft are converging toward each other. If the two aircraft are diverging (i.e., their CPA already happened somewhere in the past), the resultant t_{CPA} is negative and the initial distance between them is checked to determine any potential loss of separation (at $t_{\text{CPA}} = 0$). If the velocities of the two aircraft are identical, their distance is constant and the initial distance is used for determination of loss of separation (at $t_{\text{CPA}} = 0$).

2.2. Ownship's Maneuver

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 returning 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

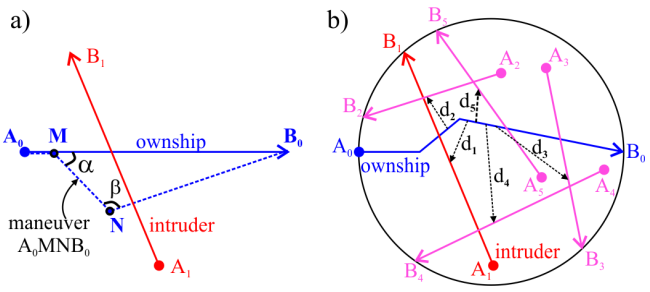


Figure 3: a) An example of maneuver. The ownship makes a heading change α° at point **M** at $t = t_1$, continues in the new heading **MN** during t_2 seconds, and heading back towards original end point at returning point **N**. A maneuver is fully defined by a set of three parameters (t_1, α, t_2).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

maneuver that satisfies $t_1 < t_{\text{CPA}}$, $\|\alpha\| \leq 60$ degrees, and t_2 takes a value such that the returning point **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.

2.3. 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 ATCOs 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. This is

further explain as follows:

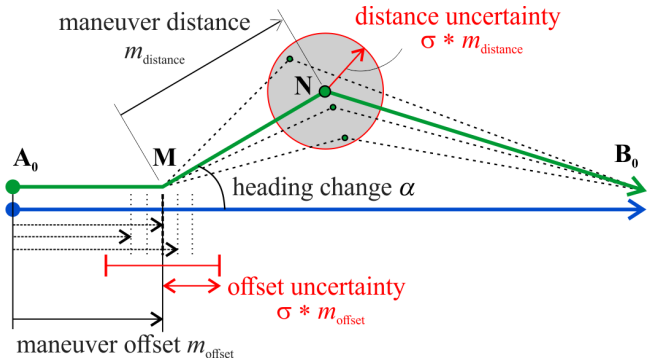


Figure 4: Environmental uncertainty and its impact on the agent's action. Three main variables for the uncertainty are maneuver offset m_{offset} and maneuver distance $m_{distance}$ and the given uncertainty level σ. The offset uncertainty affects the position of the heading change point **M**, while the distance uncertainty affects the position of the returning point **N**.

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 m_{offset} is the expected maneuver offset, α the expected heading change, and m_{distance} the expected maneuver distance. The ownship is expected to change its heading at the expected heading change point **M** and turn back at the expected returning point **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 $m'_{\text{offset}} = m_{\text{offset}} + \mathcal{N}(0, \sigma * m_{\text{offset}})$, where $\mathcal{N}(0, \sigma * m_{\text{offset}})$ is a Gaussian noise with zero mean and variance $\sigma * m_{\text{offset}}$. Similarly, the actual returning point $\mathbf{N}'$ is computed as $(x', y') = (x, y) + \mathcal{N}(0, \sigma * m_{\text{distance}})$, where (x, y) are the coordinates of **N** and (x', y') of $\mathbf{N}'$. Here, the variances of the noise distributions affecting the heading change point and the returning point are controlled by $\sigma * m_{\text{offset}}$ and $\sigma * m_{\text{distance}}$, 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.

2.4. Scenarios 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 scenarios 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 status between the ownship and aircraft i is encapsulated by 8 elements: $sep(O,i) = [\mathbf{P}^{(O,i)}, \|\overrightarrow{d_i}\|, \frac{\overrightarrow{d_i}}{\|\overrightarrow{d_i}\|}, t_{CPA}, (\vec{u}-\vec{v})]$.
  - $\mathbf{P}^{(O,i)}$: x- and y- positions of the ownship at CPA (2 elements)
  - $\|\overrightarrow{d_i}\|$: CPA closure (1 element)
  - $\frac{\overrightarrow{d_i}}{\|\overrightarrow{d_i}\|}$: x- and y- directions of the CPA closure vector (2 elements)
  - t_{CPA}: Time to CPA (1 element)
  - $(\vec{u}-\vec{v})$: Relative velocity between two aircraft (2 elements)
- Directional guidance vector (3 elements) $v_D = [\overrightarrow{\mathbf{NO}}_{\text{CPA}}, \|\overrightarrow{\mathbf{NO}}_{\text{CPA}}\|]$: This vector is chosen to be $\overrightarrow{\mathbf{NO}}_{\text{CPA}}$, where $\mathbf{N}$ is the returning point and $\mathbf{O}_{\text{CPA}}$ the location at CPA of the ownship against the intruder at the beginning. We shall discuss them in the definition of reward function.
- Turning vector (the last 3 elements) $v_T = [\overrightarrow{\mathbf{MN}}, \|\overrightarrow{\mathbf{MN}}\|]$: This vector is chosen to be $\overrightarrow{\mathbf{MN}}$. It reflects the current travel from turning point $\mathbf{M}$ to returning point $\mathbf{N}$ at each searching step.

Note that the total length of the state vector depends on the maximum number of aircraft n_{max} being considered. The state vector is a fixed length vector in the form: s = [σ, $sep(O,1)$ ..., $sep(0,n_{max})$, v_D, v_T]. In scenarios with less aircraft than n_{max}, the default values are used to fill in those missing elements to maintain the fixed length vector.

2.5. 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: More positive rewards are given to maneuvers that improve the separation status, while maneuvers that worsen the situation are punished by negative rewards called penalties. 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 $R_m(s', a)$ 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 $d_{\min} = \min_i \|\overrightarrow{d_i}\|$, $(1 \leq i < n)$, as the minimum value among all the separation distances of the ownship against other aircraft. Then, the reward function is defined as

$$R_m(s', a) = \begin{cases} e^{\frac{d_{\min}}{d_{\text{sep}}} - 1} - 1, & \text{for } d_{\min} < d_{\text{sep}} \quad (1) \\ 0, & \text{for } a \text{ is an in valid maneuver} \quad (2) \\ (1 - \dfrac{\Delta D}{\Delta D_{\max}}), & \text{otherwise} \quad (3) \end{cases}$$

where ΔD denotes the deviation of the maneuver from the original ownship's trajectory, and $\Delta D_{\max}$ the maximum deviation that could occur. In the definition of the reward function, Eq. 1 punishes invalid maneuvers that cause $d_{\min} < d_{\text{sep}}$ (negative reward) and therefore fail to separate the ownship from

other aircraft. On the other hand, Eq. 3 calculates the non-negative 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. The values for the non-negative reward range linearly from 0 to 1. Less deviated maneuvers receive higher rewards, on a score scale of maximum 1. However, since the value 1 implies no deviation from the original trajectory, it does not resolve the primary conflict and must be penalized based on our formulation. Thus, it is worth to note that the reward for the best action in different conflict scenarios will vary and never achieve the maximum score 1. Our definition for maneuver's deviation as follow.

$$\Delta D = w_1 * dist(\mathbf{M}, \mathbf{O}_{\mathrm{CPA}}) + w_2 * dist(\mathbf{N}, \mathbf{O}_{\mathrm{CPA}}), \tag{4}$$

where $dist()$ yields the distance between two points. Here, Eq. 3 and Eq. 4 imply that less deviated maneuvers shorten the distances $\mathbf{MO}_{\mathrm{CPA}}$ and $\mathbf{NO}_{\mathrm{CPA}}$. 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 $\overrightarrow{\mathbf{NO}}_{\mathrm{CPA}}$ in the state representation as mentioned in section 2.4. From empirical estimation in our conflict scenarios, the value for $\Delta D_{\max}$ is approximately 100 NM.

3. The learning model of agent and Learning Mechanism

In our problem, the ultimate goal is to train the 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) [20] for our learning algorithm. The considered action has three parameters (t_1, α, t_2) with different ranges and units that makes applying standard DDPG with 3-dimensional action challenging. In previous attempt in [37], applying DDPG directly to this problem, after certain number of training

episodes, the learned agent becomes diverged and its performance drops significantly. From our investigation, one of the major reasons for that is the complexity of the action space. In this study, we overcome this issue by proposing a modification in handling action space. The action is converted from (t_1, α, t_2) into (t_1, x, y) (where (x, y) is the coordinates of returning point $\mathbf{N}$) and a 2-stage learning strategy (separating t and (x,y) into two steps) is designed to improve model performance. This approach is an learning framework which combining searching optimal t_1 while predicting the optimal returning point $\mathbf{N}(x, y)$ for each given value of t_1. We adopt DDPG as the learning algorithm for predicting optimal returning point $\mathbf{N}(x, y)$. For bench-marking, two DRL algorithms are used (DDPG and SAC) with 3-dimentional action space (t,x,y). In this section, we briefly describe our agent, show the characteristics of our 2-stage strategy using DDPG algorithm (called DDPG-2S) that make it appropriate for training the agent, and discuss the training process.

3.1. Deep Deterministic Policy Gradient (DDPG)

DDPG [20] is a model-free off-policy actor-critic algorithm, combining DPG [38] with DQN [15]. One of its main contributions is the introduction of a neural network approximating the actor's policy to deal with continuous action space. The DDPG algorithm consists of the actor network, and the critic network. In which, the actor network, $a = \mu(s)$, is a mapping from a state s to an action a and the critic network assesses the quality of the action a (taken by the actor model) by calculating the action's value $Q(s, a)$, given a scenario s.

To improve the robustness of the algorithm, *replay buffer* and *soft target update* are employed. The *replay buffer* is used to store the past experiences for batch training, and this could improve the independence of the samples used for training the two networks. *Soft target update* is another technique that stabilizes the learning (line 21 in Algorithm 1, where the update rate is controlled by τ).

Finally, DDPG algorithm allows us to conveniently control the exploration by introducing an exploratory noise policy $\mu_{\mathcal{N}} = \mathcal{N} + \mu$, where the exploratory noise $\mathcal{N}$ is Ornstein-Uhlenbeck Noise (OU noise) [20]. Figure 6 shows examples

of exploring search path during 10 steps with OU noise, governed by $(\mu_e, \theta_e, \sigma_e)$.

3.2. The DDPG-2S for Conflict Resolution

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 2.2. 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 returning 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. This approach, DDPG for 2-stage action (DDPG-2S), is the combination of Q-learning and DDPG for learning process. In the first stage, the agent determines the heading change time t that results in the heading change point $\mathbf{M}$. The decision of the heading change time t can be considered as one action and an expected value for each decision for the given scenario must be estimated. In this study, we used DDPG to estimate that value in the second stage. The second stage decides the coordinates (x, y) of the returning 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. The detail description and explanation are discussed below.

Let's denote s the conflict scenario, a the maneuvering action, and t the time to implement the maneuver from the moment the conflict was presented to the agent. Given the conflict scenario s, the total reward the agent earns for performing an action (a, t) is $R(s, a, t)$. Let $Q(s, a, t)$ be the expected value of the reward: $Q(s, a, t) = \mathbb{E}[R(s, a, t)]$. Given scenario s and action time t, the

optimal action a^* (performed at t) is defined as

$$\begin{aligned} a^* &= \text{argmax}_a \, \mathbb{E}[R(s,a,t)] \\ &= \text{argmax}_a \, Q(s,a,t). \end{aligned} \tag{5}$$

Given the conflict scenario s, let $V(s,t)$ be the value of taking the optimal action: $V(s,t) = \max_a Q(s,a,t)$. The optimal time t^* to maximize the value $V(s,t)$ is

$$t^* = \text{argmax}_t \, V(s,t). \tag{6}$$

Our ultimate goal is to maximize the expectation of the reward earned by the agent for resolving the conflict scenario s, or to find $\max_{a,t} \mathbb{E}[R(s,a,t)]$. We decompose this optimization objective into two consecutive sub-objectives: (1) find the optimal action a^* and the respective value $V(s,t)$ at all different times t chosen from a pre-defined time grid of N values ($t \in [T_1..T_N]$) (Equation 5); and (2) select a value of t from $[T_1..T_N]$ that gives maximum value of $V(s,t)$ (Equation 6). Here, the action a is the position of the returning point **N** as described in section 2.2. We solve the former sub-objective (Equation 5) using actor-critic reinforcement learning approach. In our conflict resolution problem, the possible space for the returning point is very large and continuous; therefore the employment of the Deep Deterministic Policy Gradient (DDPG) algorithm [20] as the actor-critic algorithm is justified.

The next important task is to mathematically define the intermediate action of the agent. We define an intermediate action a_i as a moving step (dx_i, dy_i), and the agent performs a sequence of intermediate actions $[a_0, a_1, ..., a_k] \equiv [(dx_0, dy_0), (dx_1, dy_1), ..., (dx_k, dy_k)]$ in search of the optimal location (x, y) of the returning point **N**. Thus, the final action that constructs the ownship's maneuver is computed as

$$a = p_0 + \sum a_i, \tag{7}$$

where p_0 is the initial position of the returning point **N** (i.e. the CPA location of the ownship against the intruder), and a_i is the i-th searching step. Here, we impose $dx_i, dy_i \leq l$ to limit the distance that the agent is allowed to displace

the returning point in each searching step (see section 4 for the value of l). The values of dx_i, dy_i and the total number of searching steps are controlled by the learning algorithm, which is elaborated in section 3.3.

In our approach, a_i is determined by the policy function $\pi(s_i)$, in which s_i was resulted from applying the previous action a_{i-1}. Thus, the searching episode in RL algorithm generates the sequence: $(s_0, a_0, r_0) \rightarrow (s_1, a_1, r_1) \rightarrow ... \rightarrow (s_D, a_D, r_D)$, and the episode terminates right after generating the last data tuple (s_D, a_D, r_D). The termination condition is $D = S_{max}$ where $S_{\max}$ is the maximum number of searching steps allowed in one episode, or $r_D \geq Scr_{\min}$ where $Scr_{\min}$ is the minimum score that is acceptable as a valid resolution. Given a scenario s, let r_i be the reward estimated after the agent performing intermediate action a_i at the i-th searching step, we have

$$r_i = R_m(s_i, p_0 + \sum_{k=0}^{i} a_k). \tag{8}$$

In this study, uncertainty is only effective at the final state when the episode has terminated, in order to reduce the computational cost and increase the applicable of this model. Therefore, after the searching stopped, the uncertainty $\mathcal{N}$ of the environment is applied on the final position of returning point $\mathbf{N}$ to re-estimate the actual final reward.

$$r_D = R_m(s_D, p_0 + \sum_{k=0}^{D} a_k + \mathcal{N}) \tag{9}$$

Our proposed 2-stage action DDPG-2S is described in Algorithm 1 below.

3.3. Learning mechanism

The interaction between the agent and the learning environment is the core mechanism for training and testing for RL as in Figure 5. The episode is designed as a searching process to locate an "acceptable resolution" (examples can be observed in Figure 7). At each step, the agent predicts the best intermediate action (dx, dy) to modify the current returning point and send it to environment. The learning environment updates the current maneuver by

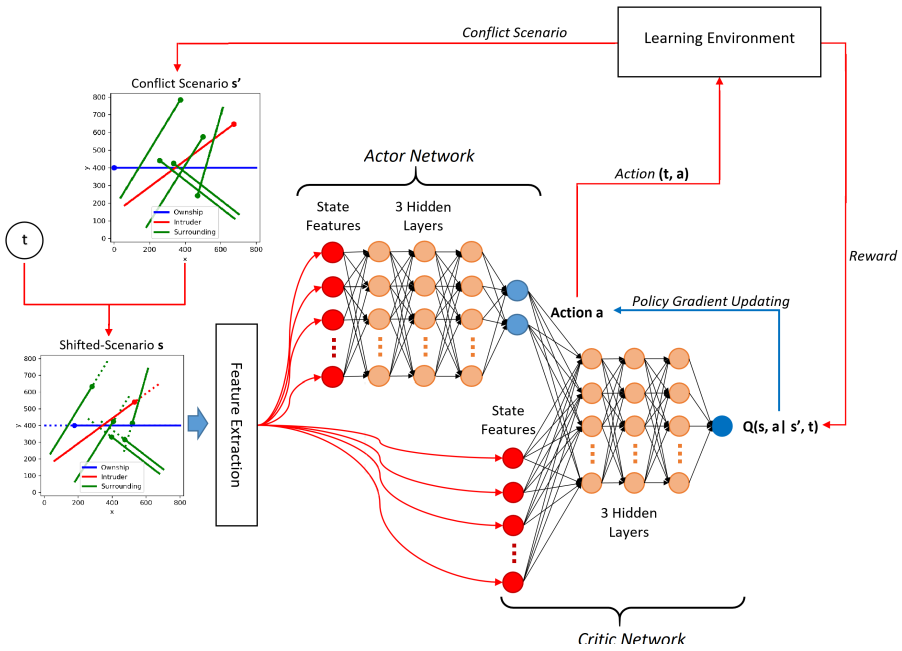


Figure 5: The proposed framework for learning conflict resolution. A conflict scenario s' from learning environment is shifted with heading change time t to obtain a conflict scenario s. Then it is used to create a feature vector for the agent. The agent includes an actor network (3 fully connected layers) and an critic network (3 fully connected layers). The red nodes are input nodes and blue ones are the output of each network. The action network is trained using policy gradient technique.

$(x', y') = (x + dx, y + dy)$, evaluates it and sends feedback back to agent. The process is repeated until an "acceptable resolution" or the maximum number of searching steps has been reached. An acceptable resolution is defined as a resolution which gains a reward greater than the minimum reward threshold Scr_{min}. After each of intermediate steps, the agent is punished by a penalty (i.e. subtract its reward by 0.1), and this is to ensure that the agent tries to reach the terminal state as quickly as possible in order to avoid such repeated penalty.

The training process is illustrated in Figure 8. The main purposes of training phase include generating learning samples to input to *replay buffer*, training actor and critic target networks using DDPG algorithm. In the interaction

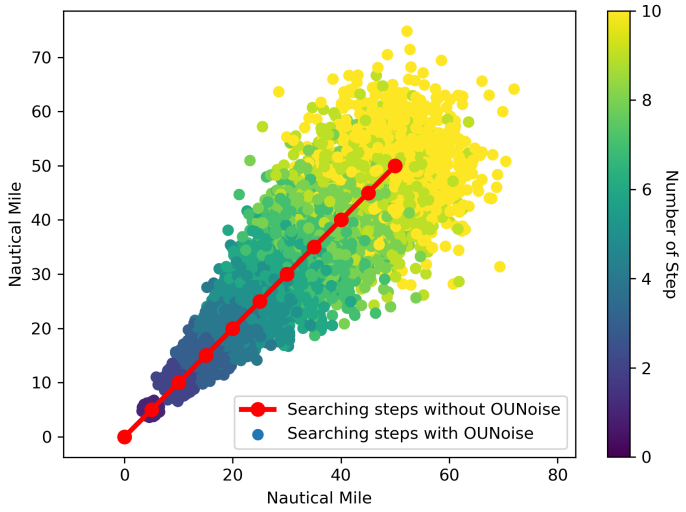


Figure 6: Visualization of exploratory search path with Ornstein-Uhlenbeck noise (OUNoise) given a constant search path (10 steps with constant step-size is 5 NM). The red line shows the expected searching path without noise, while all dots with color gradient (the number of searching steps) show 200 search paths with OUNoise.

between the agent and the environment, a set of data samples is generated and stored in the buffer. For each training iteration, a batch of sample is sampled from the buffer and used as the input to train our model, called batch training. The step by step algorithm for training is as follows.

1. Given the conflict scenario s'.
2. The scenario s' is shifted with random time duration t_0 to obtain shifted conflict scenario s_0.
3. Feature extraction algorithm is applied on Shifted conflict scenario s_i (i 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 $\mu(s_i|\theta^\pi)$ plus exploration noise $\mathcal{N}_i$) provides a candidate action $a_i \equiv (dx_i, dy_i)$.

Algorithm 1: DDPG-2S 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 $heading_change_time$ $t_0 = random(T_1, T_N)$
    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, new state s_{i+1}, status d_t
        Store transition $(s_t, a_t, r_t, s_{t+1}, d_t)$ in R
        if $d_t = True$ **then**
            Break
        end if
    end for
    Sample a minibatch of K transitions $(s_i, a_i, r_i, s_{i+1}, d_i)$ from R
    Compute expected output:

$$y_i = \begin{cases} r(i) & \text{if } d_i = 1 \\ r(i) + \gamma * Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'}) & \text{if } d_i = 0 \end{cases}$$

    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

5. The action is sent to learning environment to compute the reward r_i. If the episode termination condition is reached, the uncertainty model adds random noise to the final action and re-computes the reward r_D. Otherwise, the environment returns the intermediate constant reward $r_i = -0.1$ and updates the resultant scenario s_{i+1} after applying action to

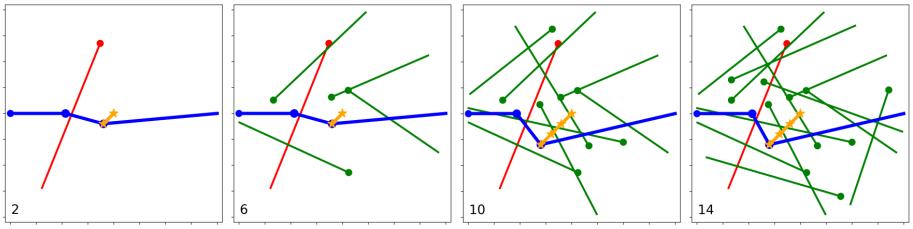


Figure 7: Examples of a searching steps to suggest resolution. From left-to-right, the same conflict with 4 levels of traffic density (2, 6, 10, 14 aircraft) are presented. The red line is the intruder, the blue one is the ownship after maneuvered and green ones are the surrounding aircraft. Yellow lines are the searching paths of the agent while the stars show the position at each searching step.

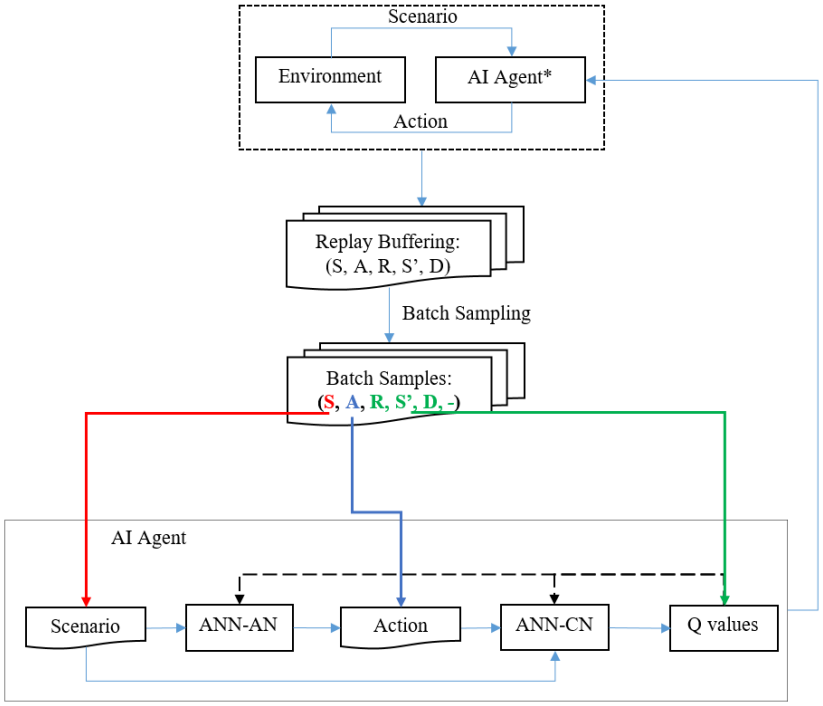


Figure 8: Training process diagram. The data generated during the interaction between the agent and the learning environment is stored in the replay buffer (S : state, A: action, R: reward, S': new state, D: Final step). A batch samples are sampled from buffer for training the agent (ANN-AN: actor network, ANN-CN: Critic Network).

modify the current maneuver.

6. The sample tuples (s_i, a_i, r_i, s_{i+1}) is stored in replay buffer for later use in training model.
7. When the replay buffer has stored sufficient 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 (line 19 in 1).
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 by soft update mechanism.
11. If end of episode is reached, the searching is terminated and the process starts over from step 1. Otherwise, increase i by 1 and return to step 4.

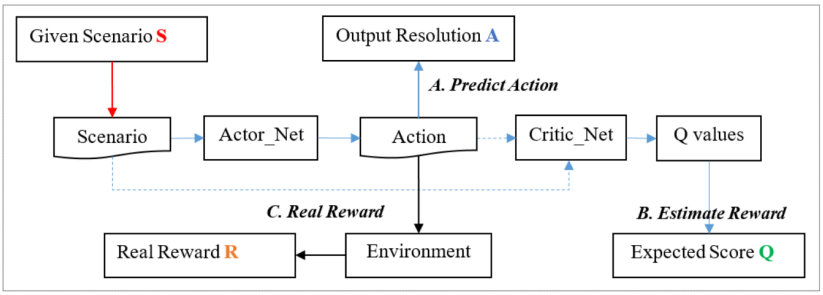


Figure 9: Testing Flowchart. Given the conflict scenario **S**, the actor network will predict the action **A** to resolve the conflict. The expected score **Q** is estimated by critic network while the real reward **R** can be obtained after applying the resolutio.

The testing phase or predicting phase is relatively simple since we only need to obtained the final recommended maneuver for a given conflict scenario (Figure 9). Only actor model is needed for this task. However, reward R and expected score Q are also computed for assessing the quality of suggested action. 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 $[T_1..T_N]$ with the increment 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 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 $V(s,t) = \max_a Q(s,a,t)$.
7. Finally, after checking with all values of heading change time t, $t^* = \arg\max_t V(s,t)$, the "optimal" maneuver for given conflict scenario s' is obtained (t^*, a^*)

Detail for training and testing processes are also described in Algorithm 1 and Figure 5.

4. Experiment Configuration

4.1. Parameters for conflict scenarios

In our experiments, conflict scenarios are randomly generated in an interested area of radius $r = 50$ nm. For the initial conflict, $d_{1(\mathrm{CPA})} < d_{\mathrm{sep}}$ where $d_{\mathrm{sep}} = 5$ nm, and $240 \leq t_{\mathrm{CPA}} \leq 480$ seconds, given that the common speed of aircraft $v_{\mathrm{c}} = 400$ 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 $n_{\max} = 30$; therefore, the state vector has fixed size of 239 (see section 2.4 for state representation). This value (30 aircraft) is also used in other conflict resolution like [28] for a given airspace with simple airway network. Different from traditional scenario where 30 aircraft will

follow predefined airways and flight levels, in our generated scenarios, aircraft don't follow any routes and all of them travel at the same flight level. Therefore, in this setting, $n_{\max} = 30$ can be considered as a high density traffic. 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, $\sigma = \{0, 2\%, 5\%, 10\%\}$.

Studies [39, 40] on conflict detection and resolution have mentioned time-to-CPA t_{CPA} and relative conflict angle ϕ (see Figure 2 for the definition of conflict angle) as important parameters. In this study, the values of those two parameters directly affect the boundary of our action space. In particular, t_{CPA} of conflict scenario constraints the possible range for selecting the heading change time t, while conflict angle ϕ implies the type of conflicts. We classify the generated conflicts into 24 groups, consisting of 4 groups of values of the time-to-CPA ($t_{\text{CPA}} \in$ [(240s, 300s], (300s, 360s], (360s, 420s], (420s, 480s]]) and 6 groups of different conflict angles ($|\phi| \in$ [(0°, 30°], (30°, 60°], (60°, 90°], (90°, 120°], (120°, 150°], (150°, 180°]]). This classification allows us to assess the model's performance in different classes of initial conflicts

4.2. Parameters for the agent

Parameters used for training the agent are shown in Table 1. The values of these parameters are chosen after performing a grid-search technique. 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. On the other extreme, $\gamma \leftarrow 1$, 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 $\gamma \leftarrow 0.99$.

4.3. Definition of Successful Resolution

Our definition for successful resolution is not only a resolution which successfully resolve a conflict without causing secondary conflict (feasible resolution), but also "high quality" or high reward. From our defined reward function, a

Table 1: The set of parameters and their values for the proposed 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	0.99
τ	Control rate of updating target networks for both models	10^{-3}
$\mu_e, \theta_e, \sigma_e$	Parameter set for exploration noise	$0, 0.1, 2$
$Scr_{\min}$	minimum acceptable reward	0.7
$S_{\max}$	Maximum number of searching steps	10
l	Upper bound of agent's searching step	5NM
Δt	Time step for heading change time	30 seconds

maneuver with positive reward is equivalent to a feasible resolution and the higher the reward is , the smaller the deviation is. In this study, the threshold = 0.7 is set as minimum reward for a successful resolution. This threshold implies that the tolerant of maneuver's deviation is 30% of the maximum deviation (approximately 30NM). Based on this definition, we define the success rate as the percentage of conflict scenarios with successful resolutions over total evaluated conflict scenarios.

4.4. Benchmark

First of all, an meta-heuristic algorithm (Tabu-Search), inspired by [13], is developed. This algorithm can achieve the optimal resolution but have high computational cost and require certain "knowledge" of environment's uncertainty for working. Thus, we use it to filtering "infeasible" conflict scenario. We are working with high-density traffic and unstructured airspace, there are cases where there aren't any feasible or good lateral resolution for ownship to resolve potential conflicts. By using the optimal algorithm to filter out such scenarios

("infeasible"), the performance of our model can be accessed more accurately. The list of conflict scenarios whose best rewards are greater than the threshold 0.7) is selected as evaluating sets. As a result, 1500 conflict scenarios are selected after this step.

Then, two RL algorithms (DDPG and SAC) are developed as conflict resolution model for comparison with our proposal approach. The implementation borrows codes from stable-baselines [41]. Noting that DDPG-2S and DDPG share architecture and hyper-parameters.

5. Results and Discussion

During the training process, the model's performance is assessed by the average reward (maximum score = 1). Figure 10 shows the smoothed curve of the three models' convergence after 300,000 iterations. In which, all models are converged (e.g., 0.85 for DDPG-2S, 0.8 for DDPG and 0.78 for SAC) after 150,000 iterations. In the training phase, it takes approximately 0.06 second for one iteration to finish (i.e., 5 hours for 300,000 iterations). In the testing phase, the agent takes approximately 0.02 second to a suggested resolution for one conflict scenario. The experimental computation was performed by a desktop PC with an Intel Xeon W-2123 CPU @ 3.60GHz, 16 gigabytes of system memory.

Figure 11 provides a closer look to the performance of the agent after convergence, indicated by the average reward and the success rate (defined in Section 4.3). To compare our proposed agent with baseline algorithms, the agent is evaluated over the set of 1500 conflict scenarios which are selected in Section 4.4. From Figure 11, we observe relations between the agent's performance indicators (i.e. score and success 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. Our model achieves a stable performance with high success rate and reward under the variation of number of aircraft (from 2 to 30) with low uncertainty ($\sigma \in (0\%, 2\%)$). It implies that the changing in the number of aircraft or the growing state-space isn't degraded

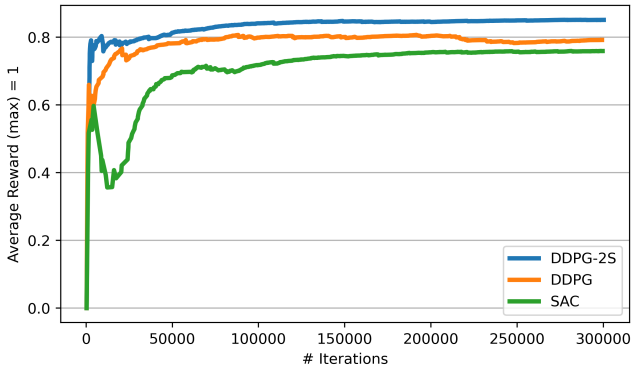


Figure 10: Illustration of the convergence of three models (DDPG-2S, DDPG and SAC) in term of average reward (Maximum = 1) during 300,000 iterations.

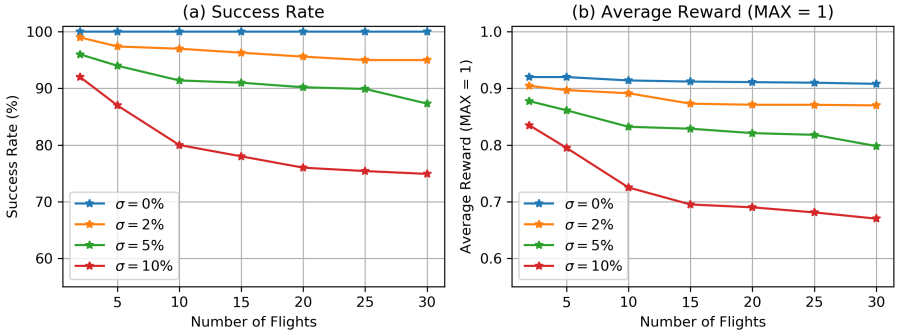


Figure 11: Average reward and success rate achieved by the agent after convergence. The performance of different traffic densities are reported (from 2 to 30 aircraft)

the model performance. When the uncertainty increases ($\sigma \in (5\%, 10\%)$), its impact on the performance becomes significantly. Besides, the results show an strong affect of increasing traffic up to 10 aircraft on the performance. From 10 to 30, the decreases in performance are insignificant comparing to lower traffic with the same uncertainty condition. Under low uncertainty ($\sigma \leq 2\%$) with any traffic density, the model consistently achieves success rate ≥ 95 and average reward ≥ 87.6. Even with medium uncertainty ($\sigma = 5\%$) the success rate is still greater than 87%. In the worst case, our model still achieves success rate

$\approx 75\%$ with the average reward ≈ 68.6.

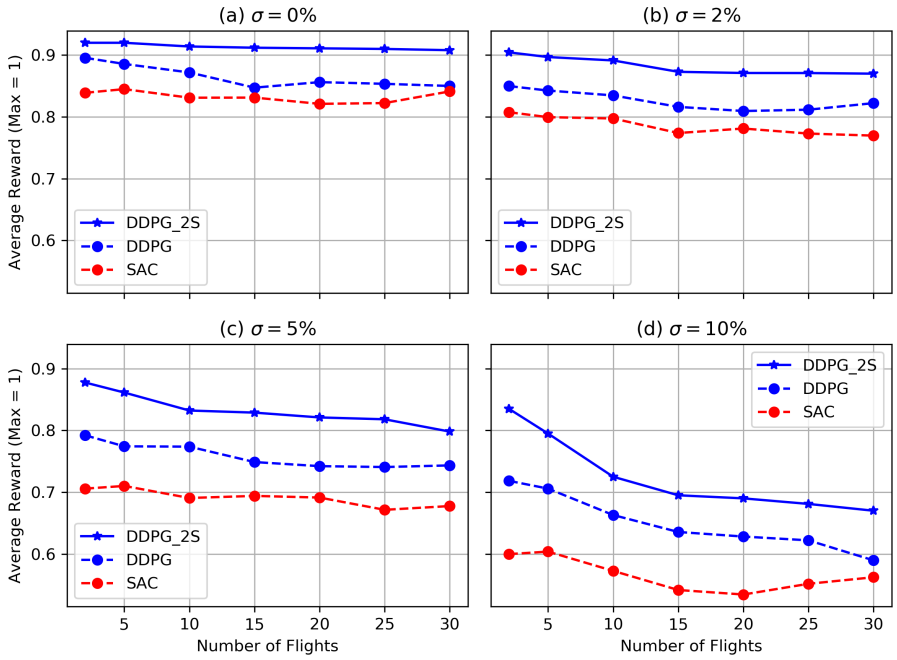


Figure 12: Comparison of average rewards between the baseline model and the proposed DDPG-2S. From left-to-right, four figures demonstrate models' performance under four different levels of uncertainty while the axis is the number of aircraft in a conflict scenario.

Figure 12 shows the experiment results of the baselines and proposed DDPG-2S model with some interesting observations. We can observe the drops in average rewards for high density traffic but it is insignificant when comparing to the impact of uncertainty level. Besides the direct affect of the success rate, the larger maneuver's magnitude is another reason for the drops. Because when the uncertainty increases, the maneuver must be larger to create more safety buffer. Besides, the high-density traffic scenarios are challenging due to multiple potential secondary conflicts. Combination of High-density traffic and uncertainty significantly increases the problem's complexity , reducing the success rate (Figure 13), as the result, the significant drops in reward can be observed. In term of individual performance, DDPG-2S outperforms both DDPG and SAC for all

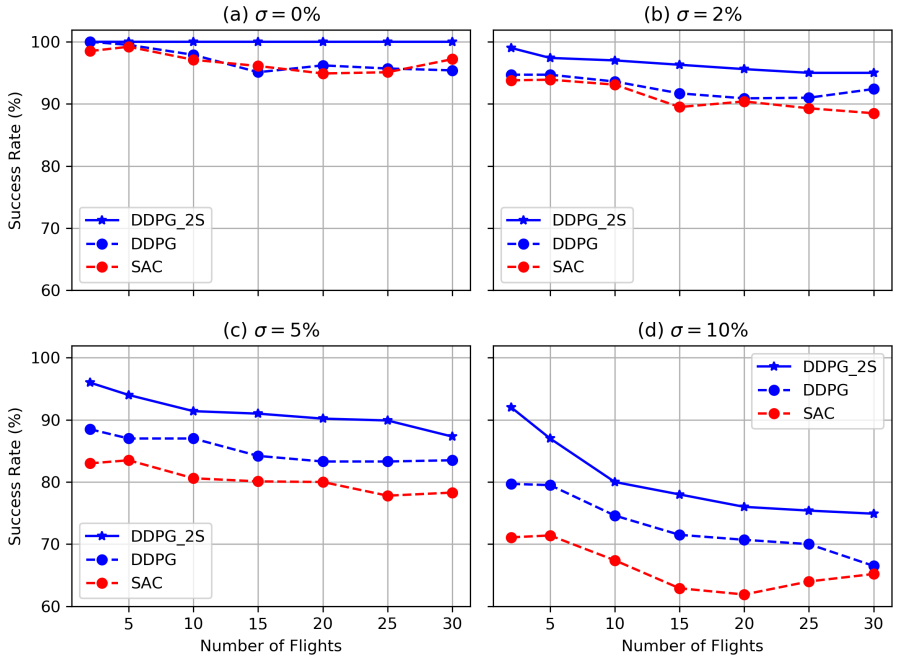


Figure 13: Comparison of success rates between the baseline model and the proposed DDPG-2S. Four figures demonstrate models' performance under four different levels of uncertainty while the axis is the number of flights in a conflict scenario

conditions. Performances of DDPG-2S and DDPG are similar with the difference within [0.05, 0.1]. Moreover, by contrasting both Figure 12 and Figure 13, the proposed model always achieves better average reward than DDPG and SAC, even for similar success rates. It implies that our model can find better solutions comparing to baseline algorithms. Even though SAC has shown better performance than DDPG in literature for high-dimensional tasks, with 3-dimensional maneuver and well-defined environment, the DDPG has shown a better result than SAC both in reward and robustness under various high-density traffic, uncertainty conditions.

The comparison on success rates is illustrated in Figure 13. The patterns, observed in this figure, are similar to ones from Figure 12. They all emphasizes the vulnerable of all models under high uncertainty. DDPG-2S still shows more

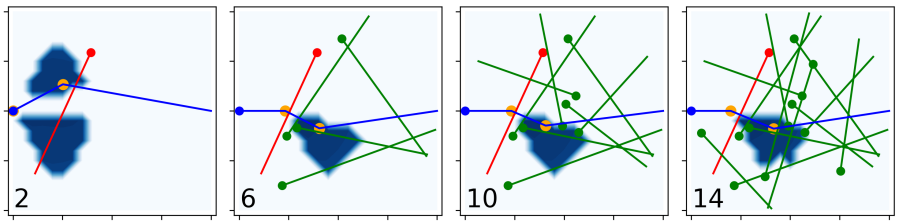


Figure 14: Examples of predicted resolutions in scenarios with different number of flights. The red lines are the intruder, the blue ones are the ownship after maneuvered and green ones are the surrounding aircraft. Two orange dots highlight the position of the heading change point and the returning point, and the color gradient shows the reward for each potential returning point.

robust performance comparing to the other two algorithms. Under medium uncertainty, DDPG-2S can maintain the success rate $\geq 87\%$.

To better understand the impact of traffic density to the difficulty of conflict resolution, some examples of conflict scenario under different traffic density are illustrated and discussed. Figure 14 presents the maneuvers suggested by the agent for resolving a conflict at different numbers of surrounding aircraft. The blue regions in the backgrounds represent the feasible regions of the maneuvers' returning points $\mathbf{N}$ (at pre-determined heading change points $\mathbf{M}$). As the number of aircraft increases, the feasible region becomes more irregular and smaller region. That change may lead to a drop in model's performance (Figure 11), especially under high uncertainty, because strong environmental disturbance highly shifts the model's suggested returning point from a feasible region to a impractical location. Figure 14 also illustrates an example of the learned strategy to resolve conflict scenario. Depending on the situation, the agent can suggests a left turn or a right turn for the heading change at $\mathbf{M}$. As shown in Figure 15, the proposed maneuvers have a quite range of turning angles from -60° to 60° with the balance in number of left turnings and right turnings. Another interesting observation is two peaks in the histogram around -15° and 15°. It reflects the strategy of the model that prefers small turning angles to maximize resolution's reward.

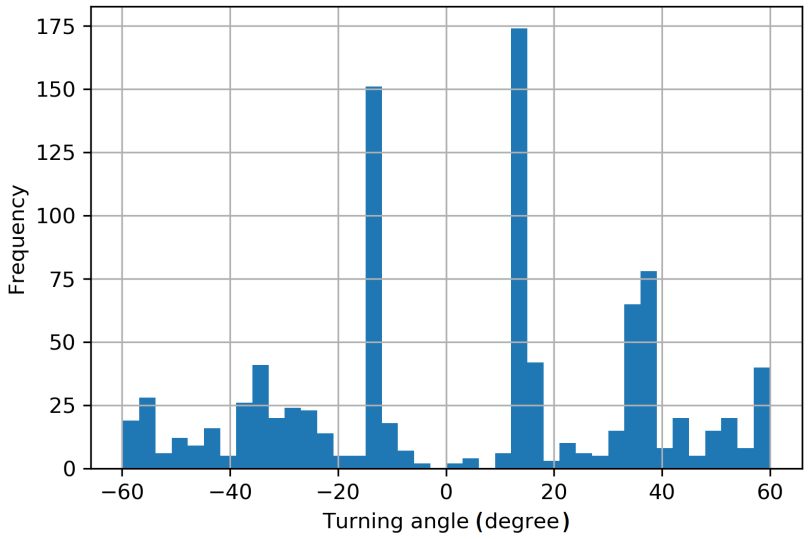


Figure 15: Distribution of proposed turning angles from our model in 1000 scenario.

6. 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 representation, reward function, and learning models, have been discussed in detail. 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. In which, the impacts of different levels of the environmental uncertainty σ and traffic density n on the model's performance are investigated and discussed. The environment uncertainty is the most important factor which affect the model's performance. Moreover, as observed in the result, the combination of high-density traffic and high uncertainty will be the challenge for any learning models.

An extension of DDPG, named DDPG-2S, is proposed to handle 2-stage ma-

neuvers under incomplete knowledge about environment. Our proposed algorithm has shown robust performance under high-density traffic and uncertainty condition. It outperforms both baseline algorithms (DDPG and SAC) in both achieved reward and success rate. Although the high uncertainty have strong impact in model's performance, DDPG-2S still obtains high success rate and average rewards. The trained model has high performance under low uncertainty level (success rate $\geq 95\%$) and medium uncertainty level (success rate $\geq 87\%$) with high traffic density. In the worst case, our model still achieves success rate $\approx 75\%$ with the average reward ≈ 68.6.

To utilize the power of RL algorithm and improve its applicability, the proposed model might be developed as an advisory tool for ATCOs by recommending possible resolution for the detected conflict scenario. For this purpose, the proposed model is designed to be suitable for working in operational environment side-by-side with ATCOs. The success rate of the model could be further improved during the interaction between the agent and the controllers. In particular, feedback from the controller, i.e. accepting or rejecting agent's resolutions, could be collected to train the agent in order to enhance its experience. However, for this purpose, a scenario transformation or mapping algorithm is needed to convert real conflict scenario to our abstracted ones. It is not a difficult task since our extracted features for the conflict scenario are generic for any conflict scenario, and will be addressed in our future work. In the event that the number of aircraft within the scenario is greater than n_{max}, the first n_{max} aircraft with smallest CPA closures could be selected as the input for the model.

Since the whole framework for training and testing agent is discussed in detail, this work can be easily adapted to different configurations and applications. One of possible configurations is considering different uncertainty settings, such as uncertainty in positions and speeds of aircraft or different uncertainty distributions. It is also possible to consider different models for trajectory prediction and probabilistic conflict detection in our framework. Another interesting adaptation is re-designing reward function for reflecting ATCOs' preferences. The

objective function could dynamically evolve to reflect the controllers' preference when the agent is exposed to new feedback data provided by ATCOs. In addition, our models could be applied for 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.

7. Acknowledgement

This research / project* is supported by the National Research Foundation, Singapore, and the Civil Aviation Authority of Singapore, under the Aviation Transformation Programme. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and the Civil Aviation Authority of Singapore.

References

[1] ICAO-Doc4444, Separation methods and minima, in: Doc 4444 Procedures for Air Navigation Services - Air Traffic Management, International Civil Aviation Organization, Quecbec, Canada, 2016, pp. 71–120.

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

[3] N. Durand, J.-M. Alliot, O. Chansou, Optimal resolution of en route conflicts, in: Procceedings of the 1st USA/Europe Seminar, 1997.

[4] Y.-J. Chiang, J. T. Klosowski, C. Lee, J. S. Mitchell, Geometric algorithms for conflict detection/resolution in air traffic management, in: Proceedings

of the 36th IEEE Conference on Decision and Control, volume 2, IEEE, 1997, pp. 1835–1840.

[5] N. Durand, J.-M. Alliot, Ant colony optimization for air traffic conflict resolution, in: ATM Seminar 2009, 8th USA/Europe Air Traffic Management Research and Developpment Seminar, 2009.

[6] J. K. Kuchar, L. C. Yang, A review of conflict detection and resolution modeling methods, IEEE Transactions on Intelligent Transportation Systems 1 (2000) 179–189.

[7] Y. Yang, J. Zhang, K. Cai, M. Prandini, Multi-aircraft conflict detection and resolution based on probabilistic reach sets, IEEE Transactions on Control Systems Technology 25 (2017) 309–316.

[8] S. Hao, S. Cheng, Y. Zhang, A multi-aircraft conflict detection and resolution method for 4-dimensional trajectory-based operation, Chinese Journal of Aeronautics 31 (2018) 1579–1593.

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

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

[11] M. Radanovic, M. A. Piera Eroles, T. Koca, J. J. Ramos Gonzalez, Surrounding traffic complexity analysis for efficient and stable conflict resolution, Transportation Research Part C: Emerging Technologies 95 (2018) 105–124.

[12] Z. Liu, K. Cai, X. Zhu, Y. Tang, Large scale aircraft conflict resolution based on location network, in: 2017 IEEE/AIAA 36th Digital Avion-

ics Systems Conference (DASC), 2017, pp. 1–8. doi:10.1109/DASC.2017.8102134.

[13] C. Allignol, N. Barnier, N. Durand, A. Gondran, R. Wang, Large scale 3d en-route conflict resolution, in: ATM Seminar, 12th USA/Europe Air Traffic Management R&D Seminar, 2017.

[14] T. Stollenwerk, B. O'Gorman, D. Venturelli, S. Mandrà, O. Rodionova, H. Ng, B. Sridhar, E. G. Rieffel, R. Biswas, Quantum annealing applied to de-conflicting optimal trajectories for air traffic management, IEEE transactions on intelligent transportation systems (2019).

[15] 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 518 (2015) 529.

[16] 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 529 (2016) 484–489.

[17] J. Schulman, P. Moritz, S. Levine, M. Jordan, P. Abbeel, High-dimensional continuous control using generalized advantage estimation, arXiv preprint arXiv:1506.02438 (2015).

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347 (2017).

[19] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in: International conference on machine learning, 2016, pp. 1928–1937.

[20] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning.,

in: International Conference on Learning Representations ICLR (Poster), 2016.

[21] Y. Duan, X. Chen, R. Houthooft, J. Schulman, P. Abbeel, Benchmarking deep reinforcement learning for continuous control, in: International Conference on Machine Learning, 2016, pp. 1329–1338.

[22] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, D. Meger, Deep reinforcement learning that matters, in: Proceedings of the AAAI conference on artificial intelligence, volume 32, 2018.

[23] B. O'Donoghue, R. Munos, K. Kavukcuoglu, V. Mnih, Combining policy gradient and q-learning, 2016.

[24] J. Schulman, X. Chen, P. Abbeel, Equivalence between policy gradients and soft q-learning, arXiv preprint arXiv:1704.06440 (2017).

[25] O. Nachum, M. Norouzi, K. Xu, D. Schuurmans, Bridging the gap between value and policy based reinforcement learning, in: Advances in Neural Information Processing Systems, 2017, pp. 2775–2785.

[26] T. Haarnoja, H. Tang, P. Abbeel, S. Levine, Reinforcement learning with deep energy-based policies, in: International Conference on Machine Learning, PMLR, 2017, pp. 1352–1361.

[27] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: International conference on machine learning, PMLR, 2018, pp. 1861–1870.

[28] M. Brittain, P. Wei, Autonomous air traffic controller: A deep multi-agent reinforcement learning approach, in: The Proceedings of the Reinforcement Learning for Real Life (RL4RealLife) Workshop in the 36th International Conference on Machine Learning, Long Beach, CA, USA, 2019.

[29] M. Brittain, X. Yang, P. Wei, A deep multi-agent reinforcement learning approach to autonomous separation assurance, Learning-based decision making for safe and scalable autonomous separation assurance (2021) 9.

[30] M. W. Brittain, P. Wei, One to any: Distributed conflict resolution with deep multi-agent reinforcement learning and long short-term memory, in: AIAA Scitech 2021 Forum, 2021, p. 1952.

[31] W. Guo, M. Brittain, P. Wei, Safety enhancement for deep reinforcement learning in autonomous separation assurance, arXiv preprint arXiv:2105.02331 (2021).

[32] J. Mollinga, H. van Hoof, An autonomous free airspace en-route controller using deep reinforcement learning techniques, in: 9th International Conference on Research in Air Transportation (ICRAT'20), 2020.

[33] S. Li, M. Egorov, M. Kochenderfer, Optimizing collision avoidance in dense airspace using deep reinforcement learning, in: 13th USA/EUROPE Air Traffic Management R&D Seminar, 2019.

[34] S. Ghosh, S. Laguna, S. H. Lim, L. Wynter, H. Poonawala, A deep ensemble multi-agent reinforcement learning approach for air traffic control, arXiv preprint arXiv:2004.01387 (2020).

[35] R. Dalmau, E. Allard, Air traffic control using message passing neural networks and multi-agent reinforcement learning, in: 10th SESAR Innovation Days, 2020.

[36] P. N. Tran, D.-T. Pham, S. K. Goh, S. Alam, V. Duong, An interactive conflict solver for learning air traffic conflict resolutions, Journal of Aerospace Information Systems 17 (2020) 271–277.

[37] D.-T. Pham, N. P. Tran, S. K. Goh, S. Alam, V. Duong, Reinforcement learning for two-aircraft conflict resolution in the presence of uncertainty, in: 2019 IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF), IEEE, 2019, pp. 1–6.

[38] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, in: International conference on machine learning, PMLR, 2014, pp. 387–395.

[39] R. A. Paielli, Empirical test of conflict probability estimation, in: USA/Europe Air Traffic Management R&D Seminar, Orlando, 1998.

[40] S. Alam, K. Shafi, H. A. Abbass, M. Barlow, An ensemble approach for conflict detection in free flight by data mining, Transportation research part C: emerging technologies 17 (2009) 298–317.

[41] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, Stable baselines, `https://github.com/hill-a/stable-baselines`, 2018.